DOCUMENT RESUME

ED 178 068 IR 007 810

AUTHOR	Harris, Diana, Ed.
TITLE	Proceedings of NECC 1979 National Educational Computing Conference (1st, Iowa City, IA, June 25-27, 1979).
PUB DATE	Jun 79
NOTE	393p.; For related documents, see IR 007 809 and ED 156 160; Legibility of some figures and examples varies; Photograph removed
AVAILABLE FROM	111 Weeg Computing Center, University of Iowa, Iowa City, Iowa 52242 ($10.00 per copy)
EDRS PRICE	MF01 Plus Postage. PC Not Available from EDRS.
DESCRIPTORS	Business Education; *Computer Assisted Instruction; *Computer Managed Instruction; *Computer Science; Engineering; Health Education; Mathematics Instruction; Special Education; Teacher Education; Teaching

ABSTRACT

The 68 presentations included in this conference proceedings discuss and evaluate educational computing, computer science, and computer engineering. This conference was held instead of the tenth Conference on Computers in Undergraduate Curricula, (CCUC) in an effort to expand the scope of information covered; the National Educational Computing Conference (NECC) is the first product of those attempts. Papers are grouped into broad subject areas, including Computer Based Education, Engineering, Pre-College Teacher Education, Computer Assisted/Managed Instruction, Minority Institutions, Natural Sciences, Computer Science, Pedagogy, Pre-College Environment, Simulation in Business and Economics, Social Science, Mathematics, Elementary School Children Programs, Micros and Minis, Handicapped, Statistics, Health, and Humanities. An author index is included. (JEG)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED178068

U S DEPARTMENT OF HEALTH,
EDUCATION & WELFARE
NATIONAL INSTITUTE OF
EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

Proceedings of

NECC 1979

National Educational Computing Conference

BEST COPY AVAILABLE

"PERMISSION TO REPRODUCE THIS MATERIAL IN MICROFICHE ONLY HAS BEEN GRANTED BY
Ted Sjoerdsma

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

Edited by
Diana Harris

Hosted by
G.P. Weeg Computing Center
The University of Iowa

25, 26, 27 June 1979

IR007810

NECC STEERING COMMITTEE

CCUC
Alfred Bork, University of California-Irvine
William Dorn, University of Denver
Gerald Engel, Old Dominion University
John Hamblen, University of Missouri-Rolla
Harry Hedges, Michigan State University
James Higgins, Digital Equipment Corporation
Glenn Ingram, National Bureau of Standards
Theodore Sjoerdsma, The University of Iowa
David Stonehill, SUNY-Binghamton

ACM/SIGCSE
Robert Stewart, Iowa State University

ACM/SIGCUE
David Kniefel, New Jersey Educational Computing Network

ACM/SIGUCC
William Knabe, The University of Iowa

ADCIS
Karen Duncan, MITRE Corporation

AEDS
Bobby Brown, The University of Iowa

AERA/SIGCAI
David Thomas, The University of Iowa

ASEE/CoED
Lawrence Jehn, University of Dayton

ECMI
Sister Patricia Marshall, Xavier University of Louisiana

IEEE Computing Society
David Rine, Western Illinois University

Society for Computer Medicine
Richard Pogue, Medical College of Georgia

LOCAL COORDINATING COMMITTEE

Chairman	Dave Thomas
Coordinator	Keith Miller
Proceedings	Diana Harris
Tours Director	Harriet Weeg

The National Educational Computing Conference wishes to thank the following people for their contribution of effort, time, and knowledge as referees for the papers submitted for presentation.

Robert Aiken, University of Tennessee
Richard Austing, University of Maryland
William Atchison, University of Maryland
R.P. Banaugh, University of Montana
Bruce Barnes, National Science Foundation
Alfred Bork, University of California-Irvine
Mary Jane Brannon, Huntingdon College
Hans Brey, Tennessee Technological University
Bobby Brown, The University of Iowa
N. John Castellan, Indiana University
Sylvia Charp, School District of Philadelphia
Ronald Collins, Eastern Michigan University
Frank W. Connolly, Montgomery College
M.K. Cueman, Virginia Institute of Marine Science
William Dorn, University of Denver
Karen Duncan, The MITRE Corporation
B.G. Dunn, Fairmont State College
Selby Evans, Texas Christian University
[illegible].T. Fink, Michigan State University
Sheldon P. Gordon, Suffolk County Community College
Keith Hall, Ohio State University
John W. Hamblen, University of Missouri-Rolla
Harry Hedges, Michigan State University
Darlene Heinrich, Florida State University
Tom Held, National Library of Medicine
James Higgins, Digital Equipment Corporation
John Hirschbuhl, Akron University
A.A.J. Hoffman, Texas Christian University
Vincent R. Jones, Southern University
Wilson Judd, McDonnell-Douglas Astronautics Co.
Martin Kamp, University of California-San Francisco
Daniel Klassen, Minnesota Educational Computing Consortium
Doris Lidtke, Towson State University
Joyce C. Little, Community College of Baltimore
Richard W. Lott, Bentley College
M.P. Lynch, Virginia Institute of Marine Science
Walter Maner, Old Dominion University
Sister Patricia Marshall, Xavier University of Louisiana
Robert Mathis, Old Dominion University
Donald H. McClain, The University of Iowa
Edmund D. Meyers, Jr., Boystown
Robert D. Montgomery, North Carolina Central University
Catherine Morgan, Montgomery County Public Schools
Amar Mukhopadhyay, The University of Iowa
Bruce E. Norcross, SUNY-Binghamton
David Pessel, University of Rochester
W.W. Porterfield, Hampden-Sydney College
James Powell, North Carolina State University
Leighton A. Price, Michigan State University
C.A. Quarles, Texas Christian University
Joe Raben, Queens College of CUNY
Ottis Rechard, University of Denver
David Rine, Western Illinois University
R.C. Rosenberg, Michigan State University
Hassell Simpson, Hampden-Sydney College

Ted Sjoerdsma, The University of Iowa
David Smith, Duke University
Philip F. Spelt, Wabash College
David Stonehill, SUNY-Binghamton
Elliot A. Tanis, Hope College
David Thomas, The University of Iowa
Robert Thompson, University of Dayton
John L. VanIwaarden, Hope College
Paul Watson, Model Secondary School for the Deaf
T.C. Willoughby, Ball State University
Gary Wittlich, Indiana University
Frank Wojcik, Virginia Institute of Marine Science
Allen Ziebur, SUNY-Binghamton

And special thanks to the authors whose cooperation kept this publication on schedule. --Diana Harris

FOREWORD

On June 16, 17, and 18, 1979 the first Conference on Computers in Undergraduate Curricula (CCUC) was held on The University of Iowa campus. It brought together over 700 participants who listened to eighty faculty members present their experiences in making the computer central to their course work. As the late Gerard P. Weeg wrote in the forward to that first conference:

> It is apparent that the computer presents our society with the possibility of making a division of effort, with those tasks which benefit by man's insight, ingenuity and instant memory recall, to be assigned to man; while those tasks which benefit from the speed, accuracy, repeatability, and indefatigability of an automaton to be assigned to one. What is more, the unfolding of computing in government, industry, and research has demonstrated that formerly unsolvable problems become solvable, only to be replaced by facets of those problems, not to mention totally new problems, which reveal themselves as being equally important and more challenging than the solved ones.

In lieu of CCUC/10 we have organized a National Educational Computing Conference (NECC) expanding the scope to include all levels of educational computing, as well as computer science and computer engineering which were previously excluded. The conference is being held in cooperation with eleven special interest groups and societies who previously held their own meetings. In a certain sense then, these proceedings represent the outcome of an experiment: bringing these special interests into a large arena to see if wider interaction stimulates new and broader activity.

It is hoped that these proceedings provide a set of papers and topics with wide interest and application to educational computing and that future conferences such as this one may have a profound beneficial effect upon the education of the future.

Ted Sjoerdsma
Chairman, Conference
Organizing Committee
University of Iowa
Iowa City, Iowa

TABLE OF CONTENTS

ADDRESS

1 Gerald L. Engel

COMPUTER-BASED EDUCATION--OVERVIEW

2 Ingredients for Excellence in Computer-Based Education Systems
Daniel E. Bailey

7 Computer-Based Education: Research, Theory, and Development
Keith A. Hall

14 The State of the Art of Instructional Computing: A Report from CONDUIT
James W. Johnson

ENGINEERING

18 On Introducing Computer-Aided Engineering Techniques into an Established Mechanical Design Curriculum
George H. Sutherland and James A.E. John

21 Computers: New Avenues for Engineering Students at the Community College
Lia Brillhart and Douglas Shawan

PRE-COLLEGE TEACHER EDUCATION

26 SIGCUE Session
Chaired by Ronald King

27 An Analysis of Computer Education Needs for K-12 Teachers
Stuart D. Milner

31 Stages of Development in Introducing Computing to Teachers
J. Richard Dennis

36 Computing Competencies for School Teachers: A Preliminary Projection for the Teacher of Computing
James L. Poirot, John W. Hamblen, James D. Powell, and
Robert P. Taylor

39 Computing Competencies for School Teachers: A Preliminary Projection for All But the Teacher of Computing
Robert P. Taylor, John W. Hamblen, James D. Powell, and
James L. Poirot

CAI/CMI

44 Computer-Managed Education: Providing Quality Education on a Large Scale
Robert J. McGlinn and Lowell A. Carmony

48 An On-Line Directory of CAI Resources
Kathleen M. Swigger and Michael Maner

52 A Successful Computerized Management System for Individualized Instruction
Gary W. Stoaks

MINORITY INSTITUTIONS

60 Computer Enhancement of Cultural Transition
Edward J. Gauss

65 Computing in Minority Institutions: 1976-1977
Sr. Patricia Marshall, Jesse Lewis, Clinton Jones, and John W. Hamblen

INVITED SESSION

84 Trends in Technology for Educational Computing
Chaired by Harry Hedges

INVITED SESSION

85 The Electronic Poet--Color Graphics Poetry: A Vision of Information
Edmund Skellings

NATURAL SCIENCES

86 Quantum Chemistry Instructional Package Using Interactive FORTRAN
Bhairav D. Joshi and James E. Eilers

92 Five Applications of Computers in Chemical Education
K. Jeffrey Johnson

INVITED SESSION

99 The Dartmouth CAUSE Report
Chaired by Bill Arms

COMPUTER SCIENCE TOPICS--I

101 A Descriptive Approach to the Teaching of Computer Organization
Zvonko G. Vranesic

107 Computer Literacy for Liberal Art Students: The Case of Computer Mapping
A. Rees Clark

INVITED SESSION

112 Authoring Systems for Developing Computer-Based Learning Materials
Chaired by Alfred Bork

PEDAGOGY

114 Computer-Assisted Test Construction via Automatic Program Generation: Using PROBGEN II to Create Individualized Exams and Problem Sets
Ronald W. Collins and Steven J. Duff

130 Computer Dialogs to Aid Formal Reasoning
Arnold Arons and Alfred Bork

COMPUTER SCIENCE--INTRODUCTORY COURSE

135 Development of a Resource Center for Introductory Computer Science
J.A.N. Lee

141 What Programming Language, If Any, for Beginners?
Herman D. Hughes

150 Initial Evaluation Results for an Introductory Programming Course Without Lectures
Margret Hazen, Cecilia Daly, David W. Embley, George Nagy, and William Prange

INVITED SESSION

160 ADCIS Session
Approaches to Lesson Distribution and Delivery
Chaired by Richard Pogue

PRE-COLLEGE ENVIRONMENT

161 A Computer Loan Service
Con Dietz and Gene A. Kemper

164 Management of Instructional Computing: Advice from Ten Pre-College Institutions
Beverly Hunter and Carol Hargan

170 OZNAKI and BEYOND
Harvey A. Cohen

SIMULATION IN BUSINESS AND ECONOMICS

179 Simulating the Great Depression in Introductory Macroeconomics
Robert Schenk

189 A Small Instructional Macroeconomic Model of the United States Economy
Frank W. Werner

194 An Examination of the Use of Computers to Increase Student Motivation and Understanding in Relation to Complex Mathematical Models
Terry L. Dennis, William J. Stevenson, and Francis J. Colella

COMPUTER SCIENCE--TOPICS II

197 The Proper Role of the Computer Science Program at a Small Undergraduate Liberal Arts University
Robert L. Horton

201 A Segmented Structure for Remedial Graduate Computer Science Instruction
Mark Pozefsky and Alan L. Tharp

205 Computer Support for Courses in Data Entry and Word Processing
Mary Jane Sunkel, Jon Todd Rickman, and Jim Hobbs

SOCIAL SCIENCE

211 Teaching Demography Through the Computer
Steven Stack

221 The Use of Census Data Analysis to Enhance the Sociology Curriculum
Harold Benenson and Steven Just

227 Comupter-Assisted Instruction for Large Classes in Economics
Donald W. Paden and Michael D. Barr

MATHEMATICS

233 Tessellations and the Tektronix 4051
Elliot A. Tanis

237 Computer Graphics as a Lecture Aid in Undergraduate Mathematics
Gerald J. Porter

ELEMENTARY SCHOOL CHILDREN PROGRAMS

250 Language in the LOGO Computer Culture
Cynthia J. Solomon

255 A Comparison of the Problem-Solving Styles of Two Students Learning LOGO: A Computer Language for Children
Daniel H. Watt

INVITED SESSION

261 Impact of Software Engineering on Computer Science Education
Chaired by Ken Magel

INVITED SESSION

262 Fundamentals of Computer Architecture and Computer Communications for Courseware Developers
Chaired by Retnadhas

MICROS AND MINIS I

264 The AVID System: A Multi-User Minicomputer-Based System for Computer-Based Education
K.W. Scholz and D.E. Sloane

268 The ISU Microsupport Net: A Prototype System for Personal Computing in Education
Richard E. Horton

271 Using Graphics for CAI with a Minicomputer
Robert T. Ward and Martin Rudnick

HANDICAPPED

274 A Computer-Assisted Instruction System for the Blind and Visually Impaired
William L. Ballenger

280 SPEDINFO: A Decision-Making Support Feature for the PLATO System
John Patrick Cronin and Peter J. Rizza, Jr.

285 A Microcomputer/Videodisc CAI System for the Moderately Mentally Retarded
Ron Thorkildsen, Warren Bickel, and Joseph G. Williams

STATISTICS

291 A Computer-Based Laboratory for General Statistics
R.S. Cunningham

297 Teaching Statistics and Using Computers
Kenneth Preskenis and Robert Oberg

HEALTH I

301 A Problem-Solving Model in Continuing Education for Dental Practitioners
Laurette F. Lipson, Kenneth D. Keith, and Gary L. Jones

311 A Training Program in Health Computer Sciences
Lynda B.M. Ellis and Laël C. Gatewood

316 Developing a Teaching/Learning Experience for Nurses in Fundamentals of Computer Programming Preliminary to Nursing Research
Kathryn Erat and Susan McGrath

HUMANITIES

326 Generative CAI for Foreign Languages
Henry W. Decker and Thomas Rice

329 Melodic Perception Development and Measurement through Computer-Assisted Instruction
Douglas Ray and Rosemary N. Killam

335 The Stanford Logic Course: Design, Dissemination, and Demonstration
Tryg A. Ager and James McDonald

MICROS AND MINIS II

343 Microcomputer Software Development: New Strategies for a New Technology
Kent T. Kehrberg

347 Microcomputer-Generated Videotapes
S.L. Cooke, Jr. and William R. Martin

HEALTH II

354 An Adverse Drug Interaction System: Its Design, Operation and Use
Edwin J. Jaufmann, Jr., Robert L. Williams, and Michael A. Barletta

360 Toward the Development of a Computer-Managed Instructional System in Pharmacy, Based upon Integrated Database System Concepts
James G. Henkel and Linda Wilkinson

371 Design of a CAI Tool to Generatively Teach Etymology
Alfredo Garcia and Sara Rude

FACULTY CONSIDERATIONS

378 Computer Training for the Faculty at Drew University
J.A. Metzler, G.V. Nelson, and W.J. Carroll

380 Transitioning Instructor Roles in Computer-Based Environments: A Preliminary Program for Improving Instructor Attitudes
Jacqueline Lea Dobrovolny, Barbara L. McCombs, and Wilson A. Judd

INVITED SESSION

386 CONDUIT Session
Instructional Styles for Microcomputer-Based Educational Materials
Chaired by Harold J. Peters

387 Author Index

The Education Activities of Professional Societies in the Computer Field

Gerald L. Engel
Old Dominion University

Much of the motivation for the National Educational Computing Conference came from the idea of exchanging ideas among the various groups interested in computers and education.

The range of interests in these groups is broad, covering virtually all aspects that might reflect the use or the study of computers in an educational setting. Interests range from the study of computer science and engineering to the use of the computer in a course in the humanities to the use of computers as an administrative aid in education. Levels of education to which computers are applied, and hence are of interest to the groups, range from precollege, through universities, to continuing education.

The structure of the groups is as diverse as their interests. Included are special interest groups of larger organizations (e.g. SIGCUE, SIGCSE), organizations whose primary interest is in the area of computers in education (e.g. AEDS), and groups that are not formally constituted societies but have attempted to meet a need through special activities (e.g. ECMI, CCUC). In addition to the organizations represented in the conference, there are a number of other organizations that have an interest in computers and education.

In this presentation, the activities of a number of organizations will be reviewed. Particular emphasis will be placed on areas that the organizations have in common. In addition, efforts that have been made to coordinate and consolidate activities involving computers in education will be discussed.

Ingredients for Excellence in Computer-Based Education Systems

Daniel E. Bailey
University of Colorado

The promise of computer-based education (CBE) remains largely a promise. One reason is that there are very few major commercial enterprises active in marketing services and systems in CBE today. There are many individual efforts underway, but they have not reached a stage of development to attract major marketing. A number of companies are active in education marketing with systems that are labeled educational, but that are really general purpose systems marketed in the educational context. It is not entirely incorrect to identify time-sharing made available in an educational environment as educational, but surely this is not what we mean when we speak of the promise of CBE.

The reason for paucity of commercial development of CBE systems are many, but in this paper I wish to look ahead to developments in the future. In particular, it is my intent to suggest some properties of future CBE systems that will make them commercially viable in the education market.

This paper attempts to pull together a wide variety of properties and characteristics that should be helpful in making future systems viable in the commercial market. A great deal of what I report here comes directly from my experience in attempting to import PLATO (Lyman, 1977; Paulson, 1976) into the University of Colorado, on three campuses. Many of the features described are incorporated in the PLATO system, many are not.

How Shall We Judge a CBE System?

The list of features presented in this paper is reasonably comprehensive. If all of the features were included in a system, we would have an ideal system. However, it may not be practical or reasonable to work with such ideals.

Under any specific situation, a specific CBE system will have: a) desired properties, b) acceptable properties, and c) tolerated properties. Three basic considerations will be the criteria for judging the appropriateness and quality of a CBE system: 1) The goals and intention of the developer will determine which properties are desirable, acceptable, and tolerable. 2) The resources available to the developer will determine how the properties are evaluated. For example, if the user is exploiting existing systems and can achieve benefits in the target program, then properties that might be intolerable under more favcrable conditions would be acceptable. 3) The historical and present situation will provide a context in which features of a CBE system will take on certain values. For example, it is important to consider what the CBE system is intended to supplement, replace, or enrich.

Features of a CBE System

I have selected a limited range of features to discuss, all of which are significant, but none are as fully developed as one might wish. The features discussed are: a) delivery systems, b) central hard-ware, c) terminal hardware, d) operating systems, e) author languages, f) student interface, g) instructor interface, h) communications, and i) data keeping.

Delivery Systems. Current work in CBE varies in terms of concern for delivery systems. The PLATO system, for example, integrates the development and delivery components. Most of the materials that reach the students are built on the system and use the terminal that the student uses. On the other hand, the TICCIT system (Mitre, 1976) involves development of substantial materials outside of the delivery mechanism.

Certain materials are difficult to develop, or otherwise manipulate, on the delivery mechanism in the CBE system. However, the general rule is that the greater the degree of integration between the delivery and development systems, the greater the potential for quality instruction. Developers should be able to integrate the most modern technology into CBE systems to achieve this goal. For example, the use of video digitizers and video disk under computer control and presentation of visual materials permits full on-line development of visuals in the delivery system. Cost currently makes the integration of such computer-based development and delivery mechanisms more research than application oriented, but soon microfiche, slide projectors, pre-recorded audio tapes, etc. will be antiquated nuisances.

The most important characteristic of a delivery system is the ability to edit, add, and delete materials as they are to be seen and manipulated by the student.

Central Hardware. In evaluating a computer system in CBE it is difficult to distinguish hardware and software properties. However, certain advantageous features in central hardware can be discerned. Also, certain features of general purpose computer systems facilitate implementation of CBE systems. Chief among these features are: a) type of terminal supported, b) type of languages supported, c) responsiveness of the computer in a multiuser interactive environment, d) the number of terminals supported within acceptable limits on response time, e) efficiences achieved by hardware capabilities such as swapping devices, stack operation, interrupts, re-entrancy, etc. Features such as high-speed, multichannel communications capability, character and bit-oriented hardware capability, rapid context switching capability, high-speed re-entrancy at low overhead cost in execution, very large scale memory address fields, highly overlapped access to large scale peripheral storage devices for extremely efficient transfer of information to central memory, etc. are more important to CBE than high-speed, high-precision floating point arithmetic.

There perhaps is a lesson to be learned from the development of PLATO. The machine used was the CDC 6400. These machines are very powerful, especially for numerical computation. However, they are deficient in many of the properties labeled above as desirable. Yet, the University of Illinois Computer Education Research Laboratory (CERL) PLATO system can run 400 to 600 student terminals simultaneously with limited central memory and swapping memory. The standard time-sharing systems offered commerically by CDC on the same equipment normally are capable of dealing gracefully with 1/10th that number of terminals used interactively. Some of these achievements can be attributed to the specialized equipment attached to the computer and to restrictions in types of computing undertaken. Thus, even with equipment not suited to the application, a great deal of highly satisfactory CBE can be achieved. But, even more could be achieved with appropriate hardware. The brilliance of the exploitation of the CDC 6400 in PLATO would have been more rewarding and productive had equipment with the features listed above been used.

Terminal Hardware. The type of terminal and the computer support of the terminal make the most visible impact on CBE. Four terminal characteristics are discussed: a) the type of display, b) the keyboard, c) auxiliary equipment, and d) student hard copy output.

a. Type of display. There is no greater impact on the student in CBE than the type of display involved. Early work in CBE used primitive keyboard/printers which were noisy, had limited character sets, were inflexible with output format, lacked graphics, and were painfully slow. Modern printers or typewriter-like terminals are hardly better, although they are faster, quieter, and have a more flexible format, with possibilities of alternate fonts, forward and back spacing, roll-forward and roll-back paper feed. Terminals of this type are minimally interactive, providing the user with little better than a typewriter with which to interact.

Cathode ray type (CRT) terminals are superior to the printer as student and lesson development displays. However, many of the CRT terminals are simply replacements of the typewriter-like terminals that are faster and quieter. Features such as message (block) mode communication from terminal to computer, edit control from the keyboard (cursor positioning), larger screens, and special character sets enhance CRT terminals as CBE terminals.

Sliding up the scale a bit we encounter graphics terminals with the capability of representing lines and drawings. The places where individual characters can be written is generalized beyond the fixed character locations on the character-oriented displays. In CBE applications the most desirable features are speed, selectivity in change of detail in the display, resolution of detail or presentation on the display screen, and low overhead on the computer communications facilities used. Also of great interest, but not generally implemented now, is point by point display at different levels of brightness. A system combining these features would have high resolution and be capable of change swift enough to produce high quality animation and dynamically changing displays. The data source for such displays must incorporate digital information. Integration of video images with the digitally driven display would be a large step in the right direction to be followed by full color. The development of video games, computerized and interactive, portends a rapid development of the technology to make these characteristics affordable in CBE.

b. The keyboard. Although less dramatic than the display used in CBE, the keyboard is very important. With possible exception of light pens and touch-sensitive display screens, the keyboard is the primary device through which the student interacts with the computer. Someday we may have sufficient knowledge and technical skill to use voice recognition and voice generation in the interaction. In the meantime, keyboards must carry the interactive burden. An ordinary typewriter keyboard is not sufficient. Many additional functions must be incorporated for use by the student and lesson developer. The commercial systems, especially TICCIT and PLATO, recognize these needs with special function keys that permit the student to ask for help, to back up, or to initiate special branching.

A major problem is the need for a large number of function keys, each requiring space. This has been solved by a shift key on typewriters and by shift and function keys on interactive terminals. However, the status of the function selected and active at any given time is often difficult to discern. Consequently, keyboards should carry some sort of status indicators in plain English, using LED or liquid crystal read-outs, and keys with dynamic character displays to indicate what function, character, or symbol will be selected by a keypress. Furthermore, a truly effective keyboard would incorporate simple directions on how to use the special features and how to select various modes in which the keyboard can be operated.

c. Auxiliary Equipment. Systematic efforts in developing general purpose CBE systems have incorporated facilities and equipment not normally thought of as computer terminal equipment. Slide projectors, audio tape and disk, and other electromechanical devices have been included in packaged systems. These enhancements are important in realizing the full capabilities of CBE. Most intriguing are devices that perform "intelligent" services: voice recognition, voice synthesis, music snythesis, video generation, etc. A full range of auxiliary facilities should be within the capabilities of a general CBE system. Since specific requirements will depend on the environment and the subject matter, it is not very practical to specify what auxiliary equipment should be incorporated into a terminal. Rather, it is more likely that interfaces should be developed with standards of data control and connectors for a wide range of auxiliary devices.

d. Hard Copy. It is important not to overlook the students' need for accessing their work at a later time, having reference notes, records, etc. For these purposes, hard copy devices should be available in a CBE system. Students, instructors, and lesson developers should be able to obtain printed versions of the information of the CBE program/session.

Operating Systems. On systems not dedicated to CBE, or where vendor-supplied systems are involved, the CBE applications depend on the features of the system. Some of the basic features are general properties of digital computer software. As in hardware specifications, the actual details of the operating system are not as important as the overall result in lesson development and lesson execution. Certain features are necessary to achieve efficiency and excellence.

Supporting and developing numerous lessons needs rapid context switching without loss of information. Similarly, many multi-interactive user situations can best be accommodated when the operating system supports re-entrancy of programs so that multiple processes can use copies of a single program or routine. A number of programming systems permit such re-entrancy in the basic operating systems, but fail to generalize those services to the user program. Other resources of the computer must be similarly shared. For example, the file system must permit multiple users to access and write to a single file without losing data.

The operating system under which a CBE system functions must be capable of managing time in a manner that permits sufficient numbers of terminals and processes to co-exist within a real-time period. Priorities and schedules must be managed so that responsiveness at student terminals is never so degraded as to interfere with the instructional activity. Such responsiveness is associated with hardware speed and other hardware features, but software often determines finally the adequacy of system responsiveness.

The types of devices supported by a computer system will be determined by the software available in the operating system and by the physical interfaces available. Character sets recognized and accepted, terminal control functions, terminal speeds, etc.—all have important representation in operating systems. Unless the desired features are supported in the operating system, the application may not succeed even if other conditions are satisfied. In particular, certain communication standards tend to limit the range of codes that can be interpreted by an operating system. Without support of certain codeless features (e.g. binary transmission over serial communications lines), specialized devices may not be supportable by an operating system.

Author Languages. There is, of course, a very heavy inter-dependency between author language and type of equipment used. Fundamental facilities, like those found in time-sharing systems, are necessary requirements in CBE systems: editors, file management utilities, execution command structures, etc. In addition, a number of features must be supplied for the special functions of authoring lessons. All of the equipment must be supported, including the graphics and specialized equipment such as touch panel, light pen, auxiliary equipment interface, etc. Standard commands should judge student input and tolerate linguistic irrelevancies, misspelling, and other incidental factors. The lesson author should be able to use student input text to assist in generating output to the student. The author should govern easily the collection of student performance data including detailed response records, use of function keys, test scores, etc. A wide variety of branching and program flow functions must be available. And, the logical and computational features of the most powerful programming languages should be standard features in the author language.

In addition to the author language itself, there should be high level support for authoring with special editors and composition programs. These support programs should provide the author easy to use and flexible means of developing special characters, graphic displays program sequences of sophisticated sorts, etc. Furthermore, the support systems should have utilities to aid integrating the auxiliary equipment into lessons. For example, recording and play-back of auditory material, video generation, and digitization, etc. should be aided by special author-oriented editors and utilities.

Student Interface. Most of the students' contact with the CBE program is with the CRT's terminal display and keyboard, and the auxiliary equipment discussed above. However, there are other kinds of contact. For example, log-on and log-off procedures should be simple. Unsuccessful log-on attempts should result in instructions on how to log on or obtain the information necessary.

The student should have extensive lesson selection and control provided both by the operating system and by the lesson. For systems that permit general instruction with a variety of different instructional topics, there must be procedures to control or permit student options for lesson selection. Self-documenting guides to lesson catalogs and utilities available to students should be available to students.

Within lessons, the student should have available, under both system support and under lesson support, a number of aids and performance options. The student should be able to sign off at any time and send messages to the author of the lesson or the instructor. There should always be explanatory sequences providing the student with help to progress through the lesson. Enough control to allow advancing and reviewing a lesson should be provided, except when such control defeats the purpose of the lesson. The student should never be faced with dead-end lessons caused by student error, or worse still, a correct student response. In summary, the operating system components under which a student is working should have sufficient power and intelligence to prevent author and system abuses and errors from interfering with the student's work.

Instructor Interface. If a CBE is to be a general system, it must allow persons other than the system developers and the lesson authors to determine or select options of the system and lessons. First, an instructor must be able to review a lesson in a scan mode without having to duplicate student responses, problem solving, and the like, and without having to understand the detail program of the lesson. Furthermore, the instructor must be able to select from among the basic modules (like chapters and problems at the end of chapters in a text) those specific items to be assigned and determine order and criteria of performance.

An instructor must be able to select lessons, exercises, and instructional modules to formulate an instructional sequence while modifying features as dictated by the situation. Rigidity of sequencing, pacing, answer judging, etc. detracts from the utility of a lesson. Consequently, a CBE system should support procedures that permit instructors to orchestrate lessons developed by others, all without violating the copyright privileges and standards of the original lesson author.

Communications. One of the extremely attractive features of general CBE systems is inherent in the communications facilities used -- communication among author, instructor, and student. Notes and messages can easily be exchanged by providing small amounts of disk storage for student, author, and instructor, and a mechanism for addressing and transmitting notes. The depth of personal interaction between author, student, and instructor on a CBE system can thus exceed that of non-computerized instructional settings. Privacy is possible, and each communication can be personal. By allowing common access to notes and communications, public information is enhanced.

Other types of communication are also permitted when the CBE system is developed along lines sketched out above: interactive, one-on-one, and multi-user communication. A mail facility permits on-line conferencing between lesson developers, on-line advising and consulting, inspection of lesson displays by consultants during execution, etc.

Data Keeping. An instructional setting encourages keeping data on student performance, if not for grading purposes, at least for guidance and development purposes. Also, managerial interest in system activity and user performance requires that the general purpose CBE system have extensive data collection and analysis utilities.

Basic data collection should include times of log-on and log-off, lesson entry, lesson exit, and times in major activities. More extensive data collection facilities in lessons should include such items as error counters, response counters, detailed response logs, use of function keys by student, unanticipated response, etc. Lessons should have flags set to enable or disable data collection depending on the lesson segment and student status. Instructors must have options that will enable or disable data collection facilities during student use of lessons.

A collection of utilities available to authors and instructors should be available to facilitate statistical analysis and summary of data collected in the course of student activity on the CBE system.

Stand-alone Systems

A few moments reflection on the prescription above for a viable CBE system will lead the observer to conclude that stand-alone systems will not be able to satisfy a large number of the ingredients for excellence. Yet it is not uncommon to hear claims being made that microcomputer stand-alone CBE systems with cassette tapes or floppy disks will displace integrated general-purpose CBE systems. Until there has been a great deal more development in the microprocessor industry, particularly in the areas of software and mass storage, and until there develops a truly inexpensive communications network capable of carrying messages with millions of times more characters than are now being transmitted over current communications networks, the promises of microprocessor stand-alone CBE systems will remain promises.

The niche for microprocessors at the present time appears to be the basis for intelligence in terminals. These microprocessors would provide for a great deal of expansion of central facility capabilities requiring little expansion of computer equipment or power. For example, character echoing, answer judging, message buffering, timing, instrument and display control, etc. could be performed by the terminal instead of by the central computer. More terminals could be serviced and each terminal would be less affected by busy periods or computer problems at the central site.

Many of the features described above will be very difficult to implement without the use of microprocessor systems as a part of the CBE terminal. A balanced exploitation of the properties of large-scale centralized computer facilities and of the properties of microprocessor-based terminal systems is much more promising than an approach that emphasizes one or the other.

References

Paulson, R. F., Control Data PLATO System Overview. St. Paul, Minn.: Control Data Corporation, 1976.

Lyman, E. R., PLATO Highlights. Urbana, Ill.: CERL, University of Illinois, 1977.

Mitre Corporation, An Overview of the TICCIT Program. Washington, D.C.: Mitre Corporation, 1976.

Review of Computer-Based Education: Research, Theory, and Development

Keith A. Hall
The Ohio State University

This paper will present a conceptual structure of the computer-based education domain; provide references to detailed research presentations and summaries for further exploration; and point to questions and decisions made during the development of CBE materials for which a theoretical base and empirical data would be useful. The term CBE as used here includes computer-managed instruction, interactive instruction, and instructional simulation.

A Conceptual Structure of Instructional Functions and CBE Techniques

The separate research and development projects in CBE over the years can be synthesized into a structure that relates instructional functions to the instructional techniques of CBE (Figure 1). Each of the instructional techniques (management, interactive instruction, and simulation) employs the computer in a different role in the instructional process. Computer-managed instruction relies principally on the recordkeeping and summarizing power of the computer. Interactive instruction, which includes the concepts of tutorial instruction and drill and practice, presents instructional material to the learner, accepts and judges responses from the learner, provides feedback, and alters the flow of subsequent instructional material based upon the learner's responses. Instructional simulations allow students to access and manipulate pools of data without that access being precisely predetermined by an author or a programmer. These instructional techniques are arranged in an ascending hierarchy (from left to right in Figure 1) which reflects the increasing complexity of both instructional tasks and learning tasks as the learner moves through the three stages of learning (acquiring, embedding, and integrating). The locus of control shifts from the computer to the learner as the learner moves through the stages of learning. The shift from the computer to the learner is also reflected by the terms (Figure 1) used to identify the instructional functions; i.e., the instructional functions in management describe the functions performed by the computer, while the instructional functions in simulation describe functions performed by the learner. The locus of control in interactive instruction is shared by the computer and the learner and the terms describing those functions reflect the shared control.

Each succeeding instructional technique in the hierarchy requires all of the learner knowledges, skills, and behaviors provided by the preceeding stage and all of the curricular products by the preceeding stage. Therefore, it is posited that the sequence for developing CBE courseware should proceed from computer-managed instruction to interactive instruction to instructional simulation.

Useful theoretical structures and research questions often can be derived from the decision points in a process. Therefore, the predominant theme in this review is the decisions made during the process of CBE courseware development. A strategy for generating research questions is to probe, explore, and question the decisions (rather than the findings) of other researchers and developers. The reader is cautioned that decisions are usually reported only implicitly and must be deduced, inferred, and derived by the reader. The tasks of CBE courseware development lie in three major areas: 1. content design and development, 2. instructional system design and 3. computer application design. Decisions and research in these domains plus research related to the process of student learning will be presented as points of departure for further research.

Content Design and Development

The first step in content design and development is to outline the modules and lessons which will convey in topical format the scope and sequence of the course content; facilitate communications among colleagues on appropriateness of curricular planning before more detailed development commences; and provide a schematic of the course for judging and adjusting the relative emphasis to be placed on each instructional topic. Step one requires decisions related to the following

Instructional Techniques

	Management	Interactive Instruction	Simulation
Instructional Functions	Assess Diagnose Prescribe Monitor	Present Practice Feedback Consolidate Learning Enhance Retention	Apply Analyze Synthesize Integrate
General References:	Baker, 1971 Mitzel, 1974	Fisher and MacWhinney, 1976 Fletcher, 1975 Glaser, 1976 Jamison, Suppes, and Wells, 1974 Merrill and Boutwell, 1972 Robin, 1976 Vinsonhaler and Bass, 1972	Greenblat, 1973 Rosenfeld, 1975

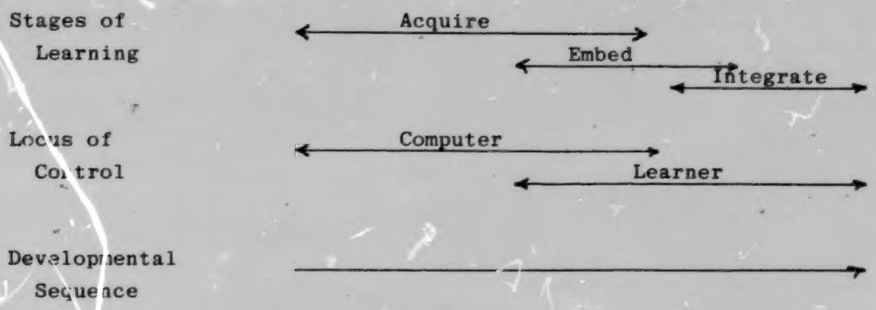

Stages of Learning
Acquire
Embed
Integrate
Locus of Control
Computer
Learner
Developmental Sequence

Figure 1. Instructional functions and requirements of CBE techniques.

questions: What are the prerequisite skills and knowledges? Do students have prerequisite skills and knowledges? Is there a societal need for providing the instruction (Posner, 1974)? What is the logical structure of the content? What sequence is appropriate -- hierarchial, simple to complex, chronological? What are the alternatives for sequencing content for instructional purposes (Posner and Strike, 1976)?

Step two is to specify the behavioral objectives which define for each lesson of each module the expected learner behaviors at the conclusion of instruction; provide advanced organizers to learners; and define learner behaviors required for successful course completion. Answer and decisions related to the following questions suggest fruitful areas for research: How detailed should behavioral objectives be -- behavior-by-behavior or encompass subgroups of behavior? Are they necessary or even desirable? Should behavioral objectives be focused on particular levels of Bloom's Taxonomy or should they span a broad range? How can they be best used with learners for maximum effectiveness (Hartley and Davies, 1976)? What are the alternatives to preparing and using behavioral objectives for structuring the courseware (Merrill, 1973)? What purposes do behavioral objectives serve (Duchastal and Merrill, 1973)?

Step three is to select the study materials for the course which will specify for each lesson of each module the extant references or instructional materials judged to be useful to the student in attaining the specified behavioral objectives. When alternative materials are available, what questions are important to answer in choosing among the options -- learner characteristics, effectiveness data, modality differences, goals of instruction (Jamison, Suppes, and Wells, 1974)? What factors should be considered when existing textual and reference materials are used for instructional purposes (Cunningham, 1972; Faw and Wallner, 1976)?

Step four is to prepare evaluation items which will evaluate for each lesson of each module the ability of each learner to exhibit the specified behavioral objectives. What types of questions are appropriate -- recall, constructed response, selection? What are the psychometric factors of alternative question types made possible by CBE? What alternative evaluation models are there for individualized instruction programs (Hambleton, 1974)? What effect does generating individual tests for each learner have on traditional statistical assumptions about measurement and evaluation? (Adams, 1974; Anderson, 1972; Hambleton, 1974; Wood, 1973)

Step five requires that criterion levels be established to define for each lesson of each module the acceptable levels(s) of student performance. What is the relationship of criterion performance during instruction to "on-the-job" performance? What alternatives are useful for evaluating non-trivial knowledges, skills, and abilities rather than trivial ones? Do adaptive versions of power tests provide new alternatives (Menges, 1975)?

Step six requires the development of new reference or instructional resources for lessons where empirical data show that extant materials do not enable learners to achieve the specified behavioral objectives. What features or characteristics of the extant resources contribute to the failure of the materials to produce learning (Merrill, 1973). What alternative forms are available for presenting content to learners (Brown and Burton, 1975)? Do information gaps provide significant improvement over other forms (Thiagarajan, 1977)? What are the relevant features contributing to the differences? What are the relationships between the form of the content in the instructional medium and the form of the content within the learner (Winn, 1977)?

Preparing an instructional simulation requires detailed specification of the initial information/conditions, role the learner will play, cause and effect relationships between all elements, the possible correct and incorrect learner decisions and the consequences of each, and a profile against which to judge learners' performances, which may include sequence(s) of steps (if important), intermediate subproblem decisions and solution(s), and final problem decisions and solutions. What variables are important in order to scale the difficulty of the case environment to the level of the learner? What empirical validation techniques can be applied to instructional simulations? What variables effect the fidelity of representation and fidelity of prediction (Crawford, 1966; Fletcher, 1971; Garvey, 1971; Greenblat, 1975; Menges, 1975; Miller, 1974)?

Instructional System Design

An instructional sequence flowchart documents all decisions about the alternative sequences of instructional events. In computer-managed instruction decisions focus on student evaluation events, criterion levels of performance, and referral to extant references and instructional materials (with computer-controlled study questions, management, and review to promote deep cognitive processing in more sophisticated applications). In interactive instruction decisions focus on alternatives for computer presentation of text, responses required of learner, and feedback to be presented to learners. In instructional simulations decisions focus on strategies for causing learners to apply, analyze, synthesize, and integrate past learning. What alternative strategies can be used to enhance learning from extant materials (Hansen, Brown, Merrill, Tennyson, Thomas, and Kribs, 1973;

Merrill, Olsen, and Coldeway, 1976; Merrill and Wood, 1975)? What factors are most significant for learning -- having learners answer questions; write questions; write summaries (Hansen, Johnson, Fagan. Tam, and Dick, 1974; Jones and Sorlie, 1976)? Should questions be general or specific (Anderson, Dalgaard, Paden, Biddle, and Surbar, 1975; Bull, 1973; Estes, 1977; Frase and Schwartz, 1975; Ladas, 1973; Wood 1973)? What alternative models are there for instructional strategies (Merrill and Wood, 1976; Tennyson, 1972; Tennyson and Merrill, 1971; White, 1973)? What factors influence the effectiveness of feedback (Kulhavi, 1977)? What alternative strategies are useful for modifying the sequence of instructional events (Lahey, Hurlock, and McGrain, 1973; White and Smith, 1974)?

A content analysis for types of learning documents the types of learning tasks included in each lesson and module and specifies appropriate instructional paradigms for those tasks. How is the acquisition of differing learning tasks effected by differing instructional strategies (Merrill, 1973)? What interactions among learners, characteristics, tasks, and instructional strategies should be studied in light of the facility via CBE to individualize each factor for each learner (Heimer, Bowers, Farris, Hirschbuhl, Hopkins, Hostetler, Pacquette, and Sawada, 1971)?

Management decisions define the rules by which a learner will traverse the domain of the case. What alternative problem-solving strategies could be used? Should the learner be carefully guided or should he be encouraged to explore? What should be the nature of feedback -- heuristic, factual statements, corrective, conceptual, specific, general? How can the quality (effectiveness) of the learner's interaction with the simulation be evaluated? What design features will cause learners to apply, analyze, synthesize, and integrate knowledge and skills? What dimensions of fidelity are critical Brown, 1977; Chartier, 1974; Koffman and Perry, 1975; Stevens and Collins, 1977; VanSickle, 1977)?

Computer Application Design

Analysis of the requirements for text presentation, response analyzing, and coding in the programming language will determine computer terminal requirements and capitalize on the power and capability of computing technology. Careful analytical study should also allow the developed techniques and products to generalize to other instructional applications. Theory, research, and development are focused in the domain of computer and information science as it relates to the processing of textual and graphic materials for instruction. Decisions require analysis of required computer tasks to determine conceptual similarities of input, processing, and output in order to facilitate computer generation of CBE courseware. Relevant research domains include artificial intelligence, decision theory, context-sensitive grammars, syntax and semantic analyses, and modeling theory. How can the effectiveness of generative CBE techniques be evaluated? How does one insure the validity and reliability of generated (as opposed to selected) instructional text? What alternative techniques can be employed to generate meaningful feedback to learner's responses (Brigham, Halverson, and Zimmerman, 1976; Elithorn and Jones, 1973; Feigenbaum and Feldman, 1963; Koffman, 1973; Schuyler, 1976)?

Courseware Use

Modified learner behavior, the *sine qua non* of the entire instructional process, is conditioned by individual differences in perceiving, cognizing, and responding. Theory, research, and development are in the domain of educational psychology. For CBE applications, particular attention should be paid to the implicit decisions and alternatives from studies of individual learners since CBE can adapt courseware to the learning characteristics of individuals (Atkinson and Shiffrin, 1968; DiVesta, 1973; DiVesta, 1975; DiVesta, Sanders, Schultz, and Weener, 1971abc; Fletcher, 1975; Greeno and Bjork, 1973; Sanders, DiVesta, and Gray, 1972).

REFERENCES

Adams, E. N. On scoring a mastery learning system control test. *Journal of Computer-Based Instruction*, 1974, 1, 2, 50-58.

Anderson, R. C. How to construct achievement tests to assess comprehension. *Review of Educational Research*, 1972, 42, 2, 145-170.

Anderson, R. C. and Biddle, W. B. On asking people questions about what they are reading. In G. Bower (Ed.), *Psychology of Learning and Motivation* (Vol. 9). New York: Academic Press, 1975.

Anderson, T. H., Anderson, R. C., Dalgaard, B. R., Paden, D. W., Biddle, W. B., and Surber, J. R. An experimental evaluation of a computer-based study in management systems. *Educational Psychologist*, 1975, 11, 184-190.

Atkinson, R. C. and Shiffrin, R. M. Human memory: A proposed system and its control processes. *The Psychology of Learning and Motivation* (Vol. 2). New York: Academic Press, 1968.

Baker, F. B. Computer-based instructional management systems: A first look. *Review of Educational Research*, 1971, 41, 1, 51-70.

Brigham, C. R., Halverson, J. D., and Zimmerman, J. QUEST: A teaching program driver. *Journal of Computer-Based Instruction*, 1976, 3, 2, 42-50.

Brown, J. S. Uses of artifical intelligence and advanced computer technology in education. In N. R. J. Seidel and M. L. Rubin (Eds.), Computers in Communications: Implications for Education. New York: Academic Press, 1977.

Brown, J. S. and Burton, R. R. Multiple representations of knowledge for tutorial reasoning. Cambridge, Mass.: Bolt, Beranek, and Newman, 1975.

Bull, S. G. The role of questions in maintaining attention to textual material. Review of Educational Research, 1973, 43, 1, 83-87.

Crawford, M. P. Dimensions of simulation. American Psychologist, 1966, 21 8, 788-796.

Chartier, M. R. Toward a theoretical model of learning as it relates to simulation games with discussion (ERIC Document Reproduction Service No. ED 088 417). 1974.

Cunningham, D. J. The retention of connected discourse: A review. Review of Educational Research, 1972, 42, 1, 47-71.

DiVesta, F. J. Theory and measures of individual differences in studies of trait by treatment interaction. Educational Psychologist, 1973, 10, 2, 67-75.

DiVesta, F. J. Trait-treatment interactions, cognitive processes, and research on communication media. Audio-Visual Communication Review, 1975, 23, 2, 185-196.

DiVesta, F. J., Sanders, N. M., Schultz, C. B., and Weener, P. D. Instructional strategies: Multivariable studies of psychological processes related to instruction. Advanced Research Projects Agency: Semi-Annual Report, July 1971a.

DiVesta, F. J., Sanders, N. M., Schultz, C.B., and Weener, P. D. Instructional strategies: Multivariable studies of psychological processes related to instruction. Advanced Research Projects Agency: Semi-Annual Report, July 1971b.

DiVesta, F. J., Sanders, N. M., Schultz, C. B., and Weener, P. D. Instructional strategies: Multivariable studies of psychological processes related to instruction. Advanced Research Projects Agency: Semi-Annual Report, Part II, July 1971c.

Duchastel, P. C. and Merrill, P. F. The effects of behavioral objectives on learning: A review of empirical studies. Review of Educational Research, 1973, 43, 1, 53-69.

Elithorn, A. and Jones, D. (Eds.) Artificial and human thinking. San Francisco: Jossey-Bass, 1973.

Estes, C. A. Effects of review quizzes as a function of density and frequency of questions during computer-managed study of text (Doctoral dissertation, The Pennsylvania State University, 1977).

Faw, H. W. and Waller, T. G. Mathemagenic behaviors and efficiency in learning from prose materials: Review, critique, and recommendations. Review of Educational Research, 1976, 46, 4, 691-720.

Feigenbaum, E. A. and Feldman, J. Computers and thought. New York: McGraw-Hill, 1963.

Fisher, K. M. and MacWhinney, B. Audio-visual autotutorial instruction: A review of evaluative research. Audio-Visual Communication Review, 1976, 24, 229-261.

Fletcher, J. D. Computer applications in education and training: Status and trends (NPRDC TR 75-32). San Diego Naval Base, Calif.: Navy Personnel Research and Development Center, April 1975.

Fletcher, J. D. Modeling the learner in computer assisted instruction. Journal of Computer-Based Instruction, 1975, 1, 4, 118-126.

Fletcher, J. L. The effectiveness of simulation games as learning environments: A proposed program of research. Simulation and Games, 1971, 2, 425-454.

Frase, L. T. and Schwartz, B. J. Effect of question production and answering on prose recall. Journal of Educational Psychology, 1975, 67, 638-635.

Garvey, D. N. Simulations: A catalogue of judgments, findings, and hunches. In P. J. Tansey (Ed.), Educational Aspects of Simulation. London: McGraw-Hill, 1971.

Glaser, R. Components of a psychology of instruction: Toward a science of design. Review of Educational Research, 1976, 46, 1, 1-24.

Greenblat, C. S. Teaching with simulation games: A review of claims and evidence. In C. S. Greenblat and R. D. Duke (Eds.) Gaming-Simulation: Rationale, Design, and Applications. New York: John Wiley and Sons, 1975.

Greeno, J. G. and Bjork, R. A. Mathematical learning theory and the new "mental forestry". Annual Review of Psychology, 1973-24, 81-116.

Hambleton, R. K. Testing and decision-making procedures for selected individualized instructional programs. Review of Educational Research, 1974, 44, 4, 371-400.

Hansen, D. H., Brown, B. R., Merrill, P. F., Tennyson, R., Thomas, D. B., and Kribs, H. D. The analysis and development of an adaptive instructional model for individualized technical training: Phase I (Technical Report AFHRL-TR-72-50 (1)). Lowry Air Force Base, Denver, Colo.: Air Force Human Resources Lab Technical Training Division, August, 1973.

Hansen, D. N., Johnson, B. F., Fagan, R. L., Tam, P., and Dick, W. Computer-Based adaptive testing models for the air force technical training environment: Phase I (Technical Report AFHRL-TR-74-48). Lowry Air Force Base, Denver, Colo.: Air Force Human Resources Lab Technical Training Division, July, 1974.

Hartley, J. and Davies, I. K. Pre-Instructional strategies: The role of pre-tests, behavioral objectives, overviews, and advance organizers. *Review of Educational Research*, 1976, 46, 239-265.

Heimer, R. T., Bower, R. G., Farris, D.C., Hirschbuhl, J. G., Hopkins, L. V., Hostettler, R. P., Pacquette, G. A., and Sawada, D. *A study of paradigms for the construction and evaluation of CAI curricular materials in mathematics: Final Report*. University Park, Penn.: The Pennsylvania State University, 1971. (ERIC Document Reproduction Service No. ED 064 163).

Jamison, D., Suppes, P., and Wells, S. Effectiveness of alternative instructional media. *Review of Educational Research*, 1974, 44, 1, 1-67.

Jones, L. A. and Sorlie, W. E. Increasing medical school performance with an interactive, computer assisted appraisal system. *Journal of Computer-Based Instruction*, 1976, 2, 3, 57-62.

Koffman, E. B. Design techniques for generative computer-assisted instructional systems. Institute of Electronic and Electrical Engineers, *Transactions on Education*, 1973, 16, 4, 182-189.

Koffman, E. B. and Perry, J. *An intelligent CAI monitor and generative tutor: Final report*. Storrs, Conn.: University of Connecticut, 1975. (ERIC Document Reproduction Service No. ED 136 826).

Kulhavi, R. W. Feedback in written instruction. *Review of Educational Research*, 1977, 47, 2, 211-232.

Ladas, H. The mathemagenic effects of factual review questions on the learning of incidental information: A critical review. *Review of Educational Research*, 1973, 43, 1, 71-82.

Lahey, G. F., Hurlock, R. E., and McGrain, P. H. *Post lesson remediation and student control of branching in computer based training* (Report SRR 73-19). San Diego Naval Base, Calif.: Navy Personnel and Training Research Lab, April, 1973.

Menges, R. J. Assessing readiness for professional practice. *Review of Educational Research*, 1975, 45, 2, 173-207.

Merrill, M. D. Content and instructional analysis of cognitive transfer tasks. *Audio-Visual Communication Review*, 1973, 21, 1, 109-125.

Merrill, M. D. and Boutwell, R. C. *Instructional development: Methodology and research* (Working Paper No. 33, Dept. of Instructional Research and Development). Provo, Utah: Brigham Young University, May, 1972.

Merrill, M. D., Olsen, J.B. and Coldeway, N.A. *Research support for the instructional strategy diagnostic profile*. Orem, Utah: Courseware, Inc., 1976.

Merrill, M. D. and Wood, N. D. *Rules for effective instructional strategies*. Oren, Utah: Courseware, Inc., 1975.

Merrill, M. D. and Wood, N. D. *Instructional strategies: A preliminary taxonomy*. Orem, Utah: Courseware, Inc., 1976.

Miller, C. G. *Some considerations in the design and utilization of simulators for technical training*. Lowry Air Force Base: Denver, Colo.: Air Force Human Resources Lab, August, 1974. (ERIC Document Reproduction Service No. Ed 097 919)

Mitzel, H. E. (Ed.) *An examination of the short-range potential of computer managed instruction: Conference proceedings*. University Park, Penn.: The Pennsylvania State University, 1974.

Posner, G. J. The extensiveness of curriculum structure: A conceptual scheme. *Review of Educational Research*, 1974, 44, 4, 401-407.

Posner, G. J. and Strike, K. A. A categorization scheme for principles of sequencing content. *Review of Educational Research*, 1976, 46, 4, 665-690.

Robin, A. L. Behavioral instruction in the college classroom. *Review of Educational Research*, 1976, 46, 3, 313-354.

Rosenfeld, F. H. The educational effectiveness of simulation games: A synthesis of recent findings. In C. S. Greenblat and R. D. Duke (Eds.), *Gaming Simulation: Rationale, Design, and Applications*. New York: John Wiley and Sons, 1975.

Sanders, N. M., DiVesta, F. J., and Gray, G. S. Effects of concept instance sequence as a function of stage of learning and learner strategy. *Journal of Educational Psychology*, 1972, 63, 3, 235-241.

Schuyler, J. A. Computer augmentation of the CAI design system. Journal of Computer-Based Instruction, 1976, 3, 2, 59-67.

Stevens, A. L. and Collins, A. *The goal structure of a socratic tutor*. Cambridge, Mass [illegible] Bolt, Beranek, and Newman, 1977. (ERIC Document Reproduction Service No. ED 138 297).

Tennyson, R. D. A review of experimental methodology in instructional task sequencing. *Audio-Visual Communications Review*, 1972, 20, 2, 147-159.

Tennyson, R. D. and Merrill, M. D. Hierarchical models in the development of a theory of instruction: A comparison of Bloom, Gagne, and Merrill. *Educational* Technology, 1971, 11, 9, 27-31.

Thiagarajan, S. Information mapping: A breakthrough in print media. *Audio-Visual Instruction*, 1977, September, 16-17.

VanSickle, R. L. *Factors affecting intellectual skill performance in instructional simulation games*. Presented at the annual program of the American Educational Research Association, April, 1977.

Vinsonhaler, J. and Bass, R. A summary

of ten major studies of CAI drill and practice. Educational Technology, 1972, 12,12; 29-32.

White, A. L. and Smith, D. C. A study of the effects of varying student responsibility for instructional decisions in a CAI course. Journal of Computer Based Instruction, 1974, 1, 1, 13-21.

White, R. T. Research into learning hierarchies. Review of Educational Research, 1973, 43, 3.

Winn, W. Content structure, cognitive structure, and media. Audio-Visual Instruction, 1977, 22, 10, 39-40.

Wood, R. Response-contingent testing. Review of Educational Research, 1973, 43, 4, 529-544.

The State of the Art of Instructional Computing:
A Report from CONDUIT

James W. Johnson
CONDUIT
P. O. Box 388
Iowa City, Iowa 52240
(319) 353-5789

The following paper is condensed from the introduction to "The 1977 CONDUIT State of the Art Report." In 1976, CONDUIT commissioned experts in instructional computing in each of several academic areas -biology, chemistry, mathematics, physics, management science, geography, political science, psychology, and sociology--to write these reports. Their expert opinions were supplemented by surveys of the literature and questionnaire surveys of college departments throughout the United States. Each report describes the current level of use in departments and courses, and tells of the specific ways in which computers are, and will be, used.

INTRODUCTION

The "1977 CONDUIT State of the Art Report" reviews the use of computers in undergraduate instruction. It tells it as it is--not as some would like it to be. You may find the current state of computing in higher education to be appalling, either because it is too small or because it is too large. Others will find that the major instructional uses of computing haven't changed much since 1968, but may be surprised at the depth of use in certain areas such as business administration. Whatever your view, you should glean from this report a clear picture of the current state of computing in higher education.

The computer is used in undergraduate learning and teaching in a variety of ways ranging from the presentation of simple drills to aiding in building complex ecological models which display results graphically. Our surveys indicate that the computer is used most often in higher education where it is almost a unique tool--in complex problem-solving, simulation and gaming, inquiry of and manipulation of data bases, and in model building. These modes of use have become an integral part of higher education, and the extent of their use is limited more by factors such as finances, available technology, or existence of software than by proven effectiveness or faculty/student interest.

The use of computing to support some part of the traditional instructional process, such as lecturing, tutoring, drilling, or demonstrating, is not deeply entrenched in higher education. Of course, what we teach, both substance and methods of inquiry, is dramatically influenced by the computer, but teaching methods themselves have been only indirectly touched. A survey of computing used to supplement instruction reveals a low level of relatively crude use. Traditional computer-assisted instruction (CAI), that is drill, tutorial, or dialog modes, constitute less than 10% of computer uses in instruction in disciplines surveyed by CONDUIT.

The relative use of the computer as estimated from our survey data is shown in Table 1. The categories are general because of the fuzziness of definitions between and even within disciplines. Not surprisingly, the categories of programming, data analysis, and problem-solving ranked highest in all disciplines.

These same surveys report that faculty interest in CAI is growing, so that the use of computers in the teaching process may increase. When asked about the need for new curriculum materials, instructors placed a high priority on CAI-type units. Chemists, for example, saw a need for drill and practice routines to help students learn basic chemical facts and principles. In addition, since current use of CAI is greatest at those schools that can provide a large amount of highly interactive service, we can anticipate an increase in the use of CAI due to the recent improvements in technology.

Table 1
Type of Computer Use
(Estimated by Discipline)

	Programming, Data analysis, Problem Solving	Simulation, Gaming, Modeling	CAI, Testing	Graphics Demonstration
Biology	54%	20%	12%	14%
Chemistry	72%	8%	8%	12%
Geography	52%	16%	8%	24%
Management	63%	20%	7%	9%
Mathematics	64%	na	na	na
Physics	50%	15%	10%	10%
Psychology	68%	20%	6%	na
Sociology	87%	6%	6%	1%

(na: data not reported)

Level of Use

The current level of use of computing in collegiate instruction is modest but significant. At least 78% of the departments responding to our surveys in chemistry, geography, physics, mathematics, and management science made some use of the computer. Quite naturally, management science stands out because of the widespread use of computing in business (60% report moderate use). Biology is noteworthy because of lack of use (47% report no use). These figures should be viewed with some skepticism because we feel that computer users were more apt to respond to our surveys.

Use of computing in specific courses is another indicator of intensity. Table 2 gives a picture of course by discipline. Mathematicians and physicists introduce the computer early in the curriculum because they see computing as a necessary skill. In social sciences, the computer is introduced later as a research tool. When it is used for biology, chemistry, and management science, it is used intensively; it occupies, at most, 20% of the time a student devotes to a course. This observation is supported by most other data we have seen.

Type of Computer Resources Used

How the computer is used in undergraduate instruction depends in part on the resources available to the teacher and the student. To make use of what computing has to offer, students must have readily available interactive stations (terminals) that operate at a minimum of 30 characters-per-second, provide less than one second response time to trivial tasks, have powerful but easily used programming languages, have sufficient memory and computational power, and offer software aids for editing, solving problems, and building models. Actual lessons offered on the computer should cover important subject matter, follow established learning principles, and have been refined through several stages of use.

This is a minimum requirement for computer system resources. Others would require a graphics display, a minimum of 120 characters-per-second operating speed, a light pen or other nonkeyboard input device, a random access slide projector, and a voice response unit to the student station. They would also demand elaborately developed course materials.

Against this baseline, we can look at the current state of computer utilization and see that it is too batch, too slow,

Table 2
Course Use by Discipline

	Average Number of Courses	Median Class Size	%Time Using Computer	Level(s) of Course
Biology	3.0	25	5%	all
Chemistry	3.5	20	10%	all
Management	5.0	50	20%	all
Mathematics	1.6	--	15%	intro
Physics	1.6	--	15%	intro
Geography	2.0	20	10%	adv
Psychology	2.0	30	15%	adv
Sociology	1.5	20	10%	adv
Political Sci.	1.5	--	10%	adv

(intro=freshman-sophmore, adv=advanced undergrad, grad)

Table 3
Mode of Computer Use in Courses

	Biology	Chemistry	Management	Mathematics	Physics	Geography
Batch	46%	47%	74%	54%	38%	79%
Interactive	66%	65%	52%	76%	56%	40%

too BASIC, too restrictive, and too underdeveloped. Looking first at mode of use (batch or interactive), in Table 3 we see that interactive computing is used more than batch in all disciplines reporting except business and geography. (Several courses use both which accounts for totals of greater than 100%.) Allowing for learning situations where batch may be best, perhaps 20% of the applications, we are inclined to conclude that batch is often used not by choice but because of lack of terminals and adequate computing equipment.

Table 4 shows the computer languages used by students or used by programs that deliver instruction. FORTRAN, a batch-oriented language, and BASIC, an interactive-oriented language, dominate. (Totals greater than 100% are possible since some courses use materials involving more than one language.) These data suggest that batch processing is still heavily used. The data are distressing to some because FORTRAN and BASIC are relatively old languages, and most of these versions do not represent contemporary programming or learning theory.

Little mention is made in any of the survey results of use of techniques such as graphics, audio-response units, or computer-controlled pictorial displays. An exception is the use of the computer for control of on-line experiments in psychology (about 10% of use.)

Source of Curriculum Materials

The reports show that most materials are locally developed for local use. For example, the physics report states: "in over half the cases, the materials were actually produced by the individual instructor." The story is the same in mathematics, chemistry, and even business administration. Only in sociology, where general purpose statistical packages dominate, does widely-distributed software make an impact. Again, this condition is the result of the early stage of development of computing, but it also means that relatively unrefined materials are often in use.

THE FUTURE OF INSTRUCTIONAL COMPUTING

Comparison of our 1976-77 data to our 1973-74 data indicates an increase in the use of computing (when the data are comparable). A supporting statistic is the responses of heads of departments we surveyed. Less than 1% of department heads saw computer use decreasing, and almost 80% saw computer use in instruction increasing in the future. Where is this increase likely to occur?

First, current use is overwhelmingly "tool-oriented." In management science for example, 27% of course use was for problem-solving, 21% for simulation, and 17% for student programming. In mathematics, 64% of computer use in courses involved student programming. Clearly, the computer is not heavily used as an instructional device although simulation has gained a foothold in several disciplines.

Second, a relatively high proportion of the instructors, 20%, asked specifically for development of drill and practice types of materials. The intended use of these materials was most likely remedial.

We found almost universal agreement among instructors and department heads that computing is a powerful tool and that its use will increase in the future. However, we are inclined to believe that estimates of increased use are based more on the pervasiveness of computing in our everyday lives than on a reasoned judgement about how computing would be used in undergraduate education.

The point is that computing can and will influence education in ways that other technologies do not. Television is a passive device. It does not help students solve problems; it doesn't help researchers manage tasks; it doesn't help me prepare text. All it does is convey information, albeit sometimes quite dramatically. Since computing gets to the heart of the goals of education--learning to think, learning to solve

Table 4
Computer Languages Used in Courses

	Biology	Chemistry	Management	Mathematics	Physics
FORTRAN	51%	59%	61%	57%	55%
BASIC	35%	42%	28%	52%	50%
PL/1	8%	7%	6%	4%	3%
APL	4%	4%	4%	15%	6%

Table 5
Use of Computing by Discipline

	Biology	Chemistry	Management	Mathematics	Physics	Geography
None, or infrequent	76%	58%	9%	62%	53%	53%
Moderate, or extensive	24%	42%	91%	38%	47%	47%

problems, learning to view one's life in perspective, and learning to seek truth--it must ultimately be used in education in in ways we do not now dare to think of.

Barriers to Improving Instruction with Computing

The picture shown here is that computer use in higher education falls short, far short, of an adequate level. Looking at the data in Table 5, we see little or no computer use in 76% of the biology departments and 53% of the physics departments. (As striking as these data are, they are understated because of the greater inclination of computer users to respond to our survey.) Looking across the totals of none and infrequent, we see that except for management science, 50% of the departments surveyed make infrequent use of the computer. What are the reasons for this deficiency?

It is usually assumed that the computer is not more heavily used in higher education because of lack of proven effectiveness or because of an antipathy toward technology. Our surveys reveal that these may not be the real problems. Our society demonstrates daily that the computer is effective, and, according to surveys, instructors are aware of that. The arguments of those who feel that colleges should stress human relationships and develop human potential in an increasingly technological world have merit. Yet, in addition to helping people develop as sensitive, humane people, we also need to help students become functioning adults. Further, by harnessing technology we can give people time to be human beings. As you may expect, our surveys revealed little antipathy toward computing; not returning the form may have indicated hostility, but one does have the feeling that current objections to the use of technology are more selective than the blanket condemnations of a few years ago. These then are barriers to use of computing that we didn't find. What are those that we did find?

Table 6
Reasons for Lack of Use of Computing

Lack of training	32%
Lack of equipment	22%
Lack of funds	12%
Lack of time	12%
Lack of interest	11%
No application	10%

When department heads were asked about barriers to use of computing, they reported the reasons as shown in Table 6.

It is significant that lack of training was mentioned three times as often as lack of interest. However, one should be careful in recommending policy based upon these results. Most students of the problem of under-utilization of the computer realize that the problem is exceedingly complex. A fair assessment is that there are several factors limiting the use of computers in higher education, including lack of training, lack of appropriate equipment in place, lack of funds for equipment, for training, and for curriculum development, and lack of time for training and curriculum development.

CONCLUSIONS

The computer has become an integral part of higher education in disciplines where it is an almost unique tool for analysis and synthesis.

The computer is not widely used as a teaching aid in higher education, but interest in and use of applications such as drill and practice are growing.

The instructional use of computing is growing at a steady but modest rate in all disciplines surveyed.

A significant number of schools make little or no use of computing in undergraduate science instruction.

The use of computing in the sciences at the undergraduate level lags behind use at the graduate and industrial levels. The gap is increasing.

Instructional computing is increasingly interactive computing using the BASIC programming language. For batch service, FORTRAN is the dominant language.

Computing equipment currently available for use in instruction is barely adequate for moderate instructional use, and inadequate for extensive use.

There is little use of commercially published curriculum materials using computers. Most use involves locally produced classnotes.

Instructors see lack of instructor training in computer use, lack of equipment, and lack of time to prepare materials as the major barriers to greater use.

The best areas for federal support include faculty training and curriculum development.

On Introducing Computer-Aided Engineering Techniques Into an Established Mechanical Design Curriculum

by

George H. Sutherland
Associate Professor & ADML Director

James E. A. John
Professor & Chairman

The Department of Mechanical Engineering
The Ohio State University
206 W. 18th Ave.
Columbus, Ohio 43210

1. Introduction

Computers have been with us long enough now that most, if not all, mechanical engineering college curricula have an introductory FORTRAN (or equivalent) scientific programming course. After this course, usually taken at the freshman or sophomore level, most mechanical design programs lack a systematic computer-oriented follow-up. It would seem reasonable for digital computers to become an integral part of the design courses. The question then is just what, when, where, and how to do this integration. This paper describes some aspects of this integration at Ohio State.

2. The Situation As It Was

The Ohio State mechanical engineering curriculum places considerable emphasis on design. Besides the design-related courses like systems dynamics, machinery kinematics and dynamics, control theory, mechanical vibrations, and general laboratory studies, the OSU mechanical engineering curriculum has five separate, required undergraduate courses that specifically deal with machine design. These courses compose a series ME 560 - 564 taken by students in their junior and senior years. Historically these design courses have had a nuts-and-bolts flavor.

The basic philosophy of the design courses is to provide students with a background capability to enable them to design machines representative of good professional practice. The requirements of good professional practice are, however, becoming more complex from year to year. What might have been good practice ten years ago might now be considered superficial and insufficient. The machine designer today has many tools at his disposal, including some powerful computer-related ones, that he must be capable of using in his day-to-day work. Thus it is important that a responsible mechanical engineering department include these tools in its core curriculum instruction.

Until June 1977, there had been no formal study at Ohio State on incorporating computer-based techniques into the design curriculum. A professor could, of course, mention such techniques in his lectures (though they were not a part of the formal course syllabi) and he could assign a simple write-your-own FORTRAN homework problem. This pattern might still be continuing except for a startling event at Ohio State— the formation of the Advanced Design Methods Laboratory (ADML).

ADML was created as a direct result of the influence of forward-thinking OSU alumni and industrial advisors. It was quite clear to these people that the age of computer-aided design (and computer-aided manufacture) was definitely upon us and that industry badly needed B.Sc., M.Sc., and Ph.D. engineers with knowledge of and capabilities in the various theories and techniques associated with CAD/CAM. To satisfy this need, ADML was set up as a focus for curriculum changes, research activities, and a capital fund drive to purchase the necessary computer-related equipment.

3. The Plan

Although the OSU ME Design faculty had differing opinions about many aspects of computer-aided engineering (CAE), they generally agreed that the introduction of CAE concepts should:

a) not dilute, distract from, or eliminate any of the course instruction dealing with fundamental machine design concepts;
b) not increase the basic credit-hour requirements for a BSME; and
c) should not require the use of advanced mathematical concepts (such as tensor calculus) or advanced computer capabilities (such as a working knowledge of IBM JCL and ASSEMBLER).

Within these general restrictions modifications were to be made to the existing design series courses so as to include CAE concepts.

The next question is just what CAE concepts should an undergraduate ME student be familiar with. The following concepts were judged appropriate for the design sequence courses:

1) generally how to use an interactive time-sharing computer terminal to solve basic engineering design and analysis tasks;
2) how computer-generated graphics, and in particular the graphics generated interactively using a graphic CRT terminal, can be used in solving engineering design and analysis tasks;
3) how the finite element method can be applied to static structure, dynamic structure, heat transfer, and fluid mechanics problems associated with designing and analyzing mechanical engineering products;
4) how to acquire component test data, digitize them, combine them with other component test or analytically determined model data, and develop a system model which can be suboptimized.
5) how mathematical optimization techniques can be employed to determine design parameter values.

The incorporation of the above concepts in the core ME design sequence is the first goal of the ADML teaching program. The particular implementation involves a combination of added laboratory experiments, course content additions, and a change in the format for individual homework problems. Briefly, it was decided that the ME 560 course, which deals primarily with material science and design philosophy concepts, be left relatively unchanged. The ME 561 course, which is primarily a failure theory course, changed the most. In this course the students receive a formal introduction to the finite element method and interactive graphics. (This course is normally taken by students in the winter quarter of their junior year.) ME 562 and 563, which are machine element design courses, build on the base laid in ME 561 and introduce optimization theory with regard to component design. ME 564, the design project course, requires the students to use the computer, and other techniques learned in the preceeding design courses to solve their design project problem. Also, the resulting design drawings and bill of materials are stored on the computer, thus giving the students some experience in dealing with a design data base. The testing experience, which involves using A/D converters and special testing-oriented computer software, is assigned to a special lab testing course, ME 581.

4. The Equipment

A critical part of the ADML program is the acquisition of suitable computer hardware and software. This acquisition involved a tremendous time and dollar investment which could not have been accomplished without outside support.

First, the authors went to interview knowledgeable people in industry and other universities. These outside people had valuable insights into what course programs were relevant, what software and hardware was appropriate, and how funding for necessary hardware and software purchases might be obtained. In fact many of these industry people have proved to be a valuable asset in obtaining industry funding support. To date the federal government has not been asked for funding. The Ohio state government has provided only a one-time loan of $165,000 to be paid back in the future. Thus the primary fiscal burden for providing equipment to the laboratory has been placed on industry. Fortunately, since most corporate profits are high right now, and graduate engineers are in short supply (especially those with a good CAE background), the concept of industry supporting ADML has been a salable one.

The OSU ME class and laboratory sizes are structured so that a total of 32 interactive terminals should prove sufficient. (As a comparative statistic the OSU ME department grants about 100 BS degrees per year.) Of those 32 terminals, 16 are graphics terminals. In addition a number of graphics tablets (4), hardcopy units (4), a plotter, and a high-speed line printer are needed as additional input/output devices.

A preliminary study indicated that the existing instruction and research computer (an IBM 370/168, which has been replaced since the study was made, by an Amdahl 470V/6-11) was not capable of properly supporting the interactive activity which was planned. The extra processing capability exists in the new Amdahl computer if the jobs are run in a batch mode. Adequate support, however, is still not possible for a graphics load of 16 terminals communicating remotely at 9600 baud. Thus the decision was made to use a local minicomputer to support the graphics activities which require a high-speed interactive connection. The minicomputer would then emulate a remote batch job entry station via a dedicated line to the Amdahl. Also eight of the terminals (four graphics and four alphanumerics) would have an additional 300 or 1200 baud direct line to the Amdahl for problems most appropriately handled that way.

Probably the two most difficult hardware questions were what minicomputer to buy and what graphics terminals to buy. Minicomputer manufacturers considered were DEC, HP, DG, Prime, and IBM. Each system had its own advantages and disadvantages relative to this particular application at Ohio State. The computer finally chosen was a brand new product from DEC, the VAX 11/780. Due to limited funding, however, and the realization that ultimately a dedicated data acquisition and control and specialized graphics minicomputer would be needed, a DEC 11/60 computer was purchased to handle the initial (less than 32) group of graphics terminals. The first group of graphics terminals purchased was four Tektronix 4014-1s with the enhanced graphics module. These terminals are probably not the most cost-effective purchases for an educational institution. However, because of their omnipresence in the industries supporting ADML, they were an obvious first choice for the initial terminal purchases.

For software ADML was most fortunate in obtaining support from SDRC, an engineering consulting firm which has developed considerable software appropriate for ADML. In particular, their testing software built around the MODAMS and SABBA programs and their minicomputer-based finite element method interactive graphics preprocessor program, SUPERTAB, have been most valuable in teaching computer-aided engineering concepts to undergraduates. For teaching general finite element method concepts, the FEAP program developed by R. L. Taylor has proved most useful due to its compactness and flexibility. For general execution purposes the SDRC SUPERB program has been adequate, though public codes like SAP, NASTRAN, or SPAR would be equally suitable. The Tektronix Plot 10 software has been used for general graphics purposes. The DEC RSX-11M operating system has been more than adequate so far, mostly because of DECUS-supplied operating system extras. A plan exists to try the phone company's UNIX operating system in the near future.

5. Problems Encountered

Any time someone tries to change the way things are done, there are bound to be difficulties. Among those found in forming ADML were equipment malfunctions, incompatibility of software, delivery delays, difficulty in hiring trained personnel, and delays due to the general university bureaucracy.

Fortunately, all problems have been solvable, and both the students and industrial sponsors are satisfied with the progress to date. In fact the student enthusiasm for ADML is even greater than the author anticipated. The interest in ADML by academically poorer students seems to be as strong as that shown by the brighter students. This might be because the poorer students see CAD techniques as a valuable supplement to their own limitations. They can now correctly solve problems using the computer that they could not correctly solve by hand.

Recommendations drawn from OSU's experiences are the following:

1. Buy only standard products with common options from well-known and preferably local vendors.
2. Involve the central computer center personnel as much as possible, as soon as possible.
3. In developing specific plans try to involve as many faculty as possible. (As a corollary, avoid general discussions — they can be infinite in number resulting in little progress.)
4. Talk to as many outside people as possible before committing time and money to specific products or courses.
5. Be precisely informed as to the correct and most efficient institutional procedures for purchasing and operating computer equipment, introducing new courses, and modifying existing courses.

6. The Road Ahead

Having established modifications to the core undergraduate curriculum, the next tasks facing ADML are in the areas of undergraduate technical option courses, graduate courses, and graduate research. Activities already exist in these areas at OSU. They are not, however, organized into a systematic framework so that a student may follow a defined sequence of courses which would identify him as a CAD major. A structured set of goals and priorities must also be established for both sponsored and unsponsored graduate research in this area. The road ahead clearly involves considerable committee work as well as personal efforts. It is in these areas that the OSU ME design faculty must work closely together in order to establish Ohio State as an international center of excellence in computer-based machine design study.

COMPUTERS: NEW AVENUES FOR ENGINEERING STUDENTS AT THE COMMUNITY COLLEGE

Dr. Lia Brillhart and Dr. Douglas Shawhan
Triton College
2000 Fifth Avenue
River Grove, IL 60171

INTRODUCTION

At Triton College, a comprehensive community college, student involvement with computers as a continued, multifaceted process is considered a primary objective of the Physics and Engineering Department. To that end, the department implemented a low-cost, diverse program.

The computer uses discussed incorporate labs, algorithms, testing, individual projects, drills, graphics, simulation, and even rhetoric. Materials for the courses listed in Figure (1) require the use of the computer in some form, interactive or batch, and are sequenced so that students become acquainted with a progressively wider variety of computer capabilities.

1st Semester	2nd Semester
Engineering Lecture/Rhetoric Engineering Graphics	Physics (Mechanics)
3rd Semester	**4th Semester**
Physics (Electricity & Magnetism) Computer Programming (FORTRAN IV) Statics	Physics (Waves & Modern Physics) Dynamics Strength of Materials

Figure (1)

Two hundred fifty students per semester, evenly divided between physics and engineering, use the computer in the modes described above. Batch processing is handled by an IBM 360-40 while a Burroughs B6803 is used for the interactive mode. Students have access to seven hard-copy terminals, two CRTs and a card reader.

INITIATION

Students entering an engineering curriculum in a community college rarely have a background in computers. While some of the feeder high schools offer computer programming courses, they are generally available only to honor students, of whom Triton College sees very few. An ACT score distribution for engineering students entering in the fall of 1976 is shown in Figure (2).

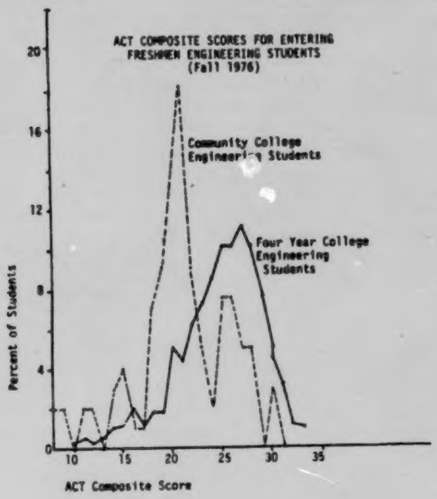


Figure (2)

The first objective is simply familiarization. Computer terminals are available in the physics/engineering complex with clear instructions for access posted. Students are shown the computer during the first week of their Engineering Lecture/Rhetoric class (a team taught combination of Introduction to Engineering and Freshman English), and are told about data processing packages available.

CALCULATIONS

The students learn to use the computer for tedious calculation work in lab, certain demonstrations, and data gathering assignments. For physics, programs exist to do linear regression curve fitting for lab data. Calculations such as mean and standard deviation can be done either by means of computer programs or by programmable calculators. Students in engineering lecture gather data from questionnaires. A computer package collates responses and reports them with mean and standard deviation as well as distribution curves. One of the most important things a student learns is that the computer is not always the appropriate place to do calculations that could be handled more rapidly with calculators.

DRILL & PRACTICE

Several computer-assisted instruction modules provide another important aspect of instruction, namely drill and practice. In physics, there are many relatively simple principles for which repeated application results in the fullest understanding. Modules using this technique include center of mass, work and energy, thermodynamics, RC circuits, DC circuits with meters. and magnetic fields due to parallel wires.

Students in Engineering Lecture/Rhetoric are given a vocabulary list that must be mastered. A computer file for comprehensive testing has been developed which satisfies two simultaneous objectives:

1. The academic objective. Students respond on a multiple choice basis and receive immediate error analysis response. Records of wrong responses are kept, and the student is responsible for retaking words missed. Furthermore, faculty can retrieve class data to determine most frequently missed words.
2. The computer-familiarization objective. Access to the vocabulary unit is simple, consisting of user and program identification. Interaction with the program is also simple, requiring only typing of the number corresponding to chosen response, waiting for evaluation (right or wrong), and pushing the return key.

In statics/dynamics drill and practice programs are used for remedial purposes. Students failing to demonstrate comprehension of certain principles must practice them on the computer before retesting is permitted. Some statics programs are three-dimensional vector calculations, location of centroids, and determination of moments of inertia.

The CAPS (Computer Administered Problem Sets) project, a significant part of drill and practice in physics at Triton College, is discussed in the next section.

HOMEWORK

CAPS is a computer-managed system at Triton College that provides individualized homework problems. CAPS was developed to modularize the assignments, provide self-pacing, assure mastery, and minimize copying and procrastination. Many experiments have shown that the best way to improve problem-solving abilities is to use small steps at a pace appropriate to each student and to provide immediate feedback. Such a modular system organizes all the instruction and insures that students cannot proceed without demonstrating competencies in needed subskills. The computer is needed to manage the system and to interact with the students whenever the students need feedback.

The CAPS project consists of two components:

1. Computer-written and printed homework.
   - (a) 20 problems are entered into a program which computes one correct answer and four distractor answers for each of 10 variations of each problem.
   - (b) The problems are printed 5 to a page with the correct answer printed randomly as one of five choices (a) through (e).
   - (c) Options include (1) multiple answer problems can be given with up to three answers per choice; (2) non-numerical answers can be given by using a coding scheme.
   - (d) Output includes 20 versions of each page in such a way that each of the 10 variations of each problem occur twice but in such a way that no two versions are identical. Three copies of each version are printed, including a file copy with answers.
2. Computer-managed and graded homework.
   - (a) Students work an assigned set of 20 problems each week. They enter their answers (a to e) into a program. The program grades the answers and indicates which ones they missed. Success for completion of the set is 80%, or 16 problems out of 20.

Percent Correct	Trial	Grade
80%	1	A
80%	2	B
80%	3	C
80%	4	D
80%	5	F

   - (b) Students may not proceed to the next set until they have correctly answered 80% of the previous set.
   - (c) To prevent procrastination, 3 sets must be completed before each of 5 tests. If all three sets are not completed prior to the test, the grades for incompleted sets become an F.

TESTING

The computer-administered testing system (CATS) at Triton has been implemented in graphics and statics/dynamics and will be implemented in physics.

Statics/dynamics and graphics are presently taught in a modularized, self-paced manner, with minimum standards of three modules per month in mechanics and one test per month in graphics required to assure completion of the course. Modules consist of objectives, lectures, solved problems, and assigned problems. Extensive use is made of audio-visual materials (slide-tape presentations, sound page, videotapes); lectures are presented when appropriate, and a tutoring service is relied upon for developmental purposes.

When a module is completed, testing is done by means of a computer. The computer both generates tests and manages test results. The student interacts with the computer as follows:

1. The computer is accessed with identification number and module (test) number.
2. The computer randomly chooses one problem of each type for that test, and prints the problems for the student.
3. The student takes the test and receives a hand-graded result back.
4. If a student fails an exam, the student must evidence further comprehension of the subject before a retest may be taken.
5. For a retest, problems are again randomly chosen from the problem file, but problems previously assigned to that student are omitted from the file.

The instructor may access a student's test answer at any time. Further, a record of attempts per test is available. Retesting is controlled by the instructor since access to the computer is denied unless further work for a module has been done.

The testing method allows students to take tests at their convenience (the testing center is open 80 hours/week). Cheating is minimized by using extensive files where each problem (not just the numbers) is different.

ALGORITHMIZATION

There are many principles in physics which are considered formal operational in the sense of the Piaget[1] model of intellectual development. Renner[2] and others have shown that a significant number of college level students operate intellectually at a concrete level. It has been shown that such students must observe physical phenomena directly by manipulating equipment themselves in order to progress to the formal operations state in the discipline. Thus, a student can be taught to deal formally with the equation F=ma by requiring that he discover the relation by means of a laboratory exercise. There are many principles of physics, however, that are primarily formal operational, i.e., there are few, if any, physical phenomena for students to directly observe to discover the principle. Algorithms, which are at a lower cognitive level than principles, provide an alternative approach to making formal operations concrete. The algorithm provides a hands-on prescription for applying the principle.

Several computer-aided modules have been developed in which algorithms are taught. Among these are units on kinematics, vectors, Gauss' Law, and Ampere's Law. Modules present the students with a series of rules for applying the principle, and the programs lead them through the various concepts with several examples, each time providing progressively fewer prompts. Then the student is tested on the use of the algorithm with a few examples for which no prompts are given. In the case of Gauss' Law and Ampere's Law, the computer instruction is totally self-paced. The principle of such a programmed system of instruction is that the student must master all subskills before proceeding to higher level skills.

SIMULATION

Currently, simulation experiments are successfully being run by students in physics, statics/dynamics, and strength of materials. The main advantage in the simulations is that they are ideal experiments. Unlike true experiments which are plagued by a variety of systematic errors, simulations demonstrate the particular principle in an errorless fashion. Generally, students vary several of the independent parameters one at a time, and see how the results are affected. Conclusions about cause and effect follow readily and students are able to deduce the correct principle. In addition, the time required for all possible variations of parameters is much less than with a true experiment.

In physics, programs run by students include projectile motion, energy conservation, collisions, electric and magnetic field plotting, DC circuits, RLC oscillations, Maxwell-Boltzmann statistics, wave interference, geometrical optics, diffraction grating, relativity, photoelectric effect, and Compton scattering. The subject of the simulation in Fig. (3) is projectile motion. For this case, the student inputs the initial speed and the angle. By selecting thirty and sixty degrees, the student readily sees that the range is the same for complements of an angle as for the angle. Also, by trying other angles around forty-five degrees, the student readily sees that the range is a maximum at forty-five degrees.

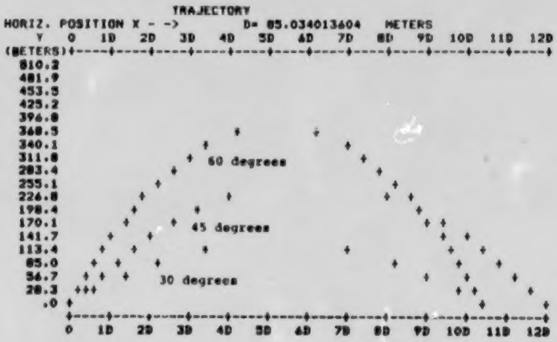


Figure (3)

In statics and dynamics simulation programs, variation of parameters clearly demonstrates phenomena. Examples are:

1. piston acceleration for various crank angles,
2. location of principle moments of inertia by rotation of axis,
3. single degree of freedom response for variation of m, c, and k,
4. forced vibrations of damped, single degree of freedom problems.

In strength of materials, students have access to a card program to calculate shear, moment, and deflection values for a straight beam under any loading and for any determinate support system. Here students are exposed to non-interactive batch processing for the first time.

PROJECTS

Students in graphics and statics/dynamics may choose a computer project to meet the requirements of the courses. For graphics they may:

1. Digitize drawings from their sketches and then manipulate them with the computer,
2. Compare drawn output against the computer output, or
3. Solve graphics problems on the computer.

When students enroll in statics/dynamics, a majority have already taken computer programming (FORTRAN IV, calculus prerequisite). For statics, students have written programs to find centroids and moments of inertia for general geometries, and for the determination of forces in trusses and the generation of equivalent systems. The student-written programs, when appropriate and educationally valid, are altered where needed and made available to all students.

VALIDATION

Extensive testing of student response has been done in physics for some of the simulations and drill and practice programs. To date the modules have been successful in increasing student achievement. A comparison was made of scores on tests of certain topics covered by CAI. Students in the experimental group had the traditional lecture and CAI as a supplement. Students in the control group had only lecture. Typical results are shown for five units in Figure (4).

	Control	Experimental
DC Circuits	51%	87%
RLC Circuits	48%	80%
Waves	70%	84%
Interference and Diffraction	62%	85%
Compton Scattering	12%	43%

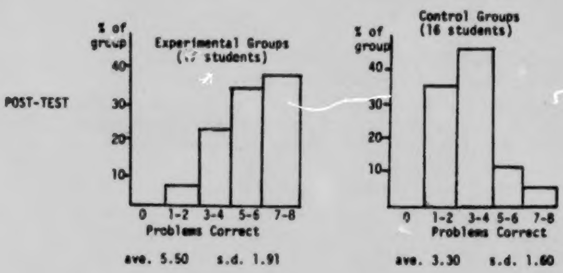
Percentages are the percentages of the class who were successful with the particular problem pertaining to the topic.

Figure (4)

The CAPS project for physics homework has been evaluated for one semester in one class. The evaluation was done by comparing the exam scores for that semester with a similar group of students who had the traditional homework problems. The average score on hourly tests increased from 65% correct to 70% correct, while on the final exam it increased from 64% to 73%. Not only did the averages increase, the standard deviation decreased; consequently, the difference in means was significant with a 95% confidence using the t-test.

For the two programs on Gauss' Law and Ampere's Law, randomly chosen students were selected for experimental and control groups. The control group received lecture, textbook, and homework problems on the two topics. The experimental group received the CAI material in addition to the lecture, textbook, and homework problems. The control group was provided with a placebo CAI problem to alleviate Hawthorne effect[3] problems. Figure (5) shows the results of the comparison of experimental groups to control groups for the Gauss' Law unit. Similar results were obtained for the Ampere's Law unit.

GAUSS' LAW



T TESTS FOR SIGNIFICANT DIFFERENCES

Pretest P1:	$t = 0.17$	not significant
Post-test T1:	$t = 3.69$	significant ($p < .01$)

Figure (5)

The data in Figure (5) indicate that the algorithmic presentation was superior to the standard lecture/recitation presentation. Students who had the special instruction performed significantly better (using the t-test) on the post-tests than the control groups.

Validation of the use of the computer in graphics and mechanics is based on a 140-item questionnaire developed by Triton College and correlation of achievement before and after introduction of the new material. Student response is positive except for major complaints about a consistently heavy work load. Students may no longer flunk a test and proceed, and they don't like that restriction. The retention rate before and after introduction of the new material is approximately the same. With the new material, however, students are automatically dropped if they do not meet the minimum pace. Student achievement has improved markedly, as evidenced by higher grades and a lower percentage of students receiving Fs.

CONCLUSIONS

The Triton College experiment introducing students to the computer has shown that a community college can effectively and at low cost implement a multifaceted program, without reliance on sophisticated programming techniques or outside expertise. The total investment in terminals is less than $9,000. All programs discussed in the paper have been developed by Triton faculty and student aids. Faculty programming has been subsidized by internal research grants and students have been paid from work study grants. Some of the material presented in this paper has been developed with the aid of a grant from the National Science Foundations (CAUSE), while other material was developed for a doctoral dissertation by Dr. Shawhan under the supervision of Dr. Ed McNeil (UICC).

While student achievement of certain objectives is enhanced using the computer-assisted instruction materials, probably more important is the student enthusiasm about computer use that has led to many voluntary computer projects.

The faculty of the Physics and Engineering departments fully believe that the computer experiment has increased academic competence of the students and has also thoroughly familiarized students with a universal tool of science and engineering.

BIBLIOGRAPHY

1. John L. Phillips, The Origins of Intellect, Piaget's Theory, (New York, 1975).
2. John W. Renner, "Significant Physics Content and Intellectual Development - Cognitive Development as a Result of Interacting with Physics Content," American Journal of Physics, Vol. 44: 218, March 1976.
3. Donald T. Campbell and Julian C. Stanley, Experimental and Quasi-Experimental Designs for Research, (Chicago, 1966).

Pre-College Teacher Education

Chaired by Ronald King
Western Illinois University

Introduction to the ACM Elementary and Secondary School Subcommittee

David Moursund, The University of Oregon

No abstract available at time of printing.

An Analysis of Computer Education Needs for K-12 Teachers

Stuart D. Milner, George Mason University
(Paper)

Stages of Development in Introducing Computing to Teachers

J. Richard Dennis, University of Illinois-Urbana
(Paper)

Computing Competencies for School Teachers: A Preliminary Projection for the Teacher of Computing

James L. Poirot, North Texas State University
(Paper)

Computing Competencies for School Teachers: A Preliminary Projection for All But the Teacher of Computing

Robert P. Taylor, Columbia University
(Paper)

Attitudes of Secondary School Teachers Toward Instructional Computing

Daniel Klassen, Minnesota Educational Computing Consortium

No abstract available at time of printing.

AN ANALYSIS OF COMPUTER EDUCATION NEEDS FOR K-12 TEACHERS

Stuart D. Milner
Department of Education
George Mason University
Fairfax, VA 22030

Abstract

This paper explicates problems and issues of computer education for K-12 teachers. In order to facilitate planning and decision making with respect to these problems, a framework for training needs is presented. The framework, based on training needs for various types of teaching, is derived from teacher training experiences and recommendations of the Association for Computing Machinery. Suggestions for implementing change in the context of the problems and issues are also presented.

Problems and Issues

It is generally acknowledged that one of the most critical problems facing effective and widespread use of computers in learning is teacher education; that is, teachers at all levels need to be educated about the use of computers in the classroom and about their impact on society in general (Milner, 1; IFIP, 2).

Unfortunately, there are numerous unresolved issues that inhibit the education of teachers at both the preservice and inservice levels. Among these are 1) lack of certification and experience requirements for teaching computer-related courses; 2) lack of educators' knowledge of computer applications in education; 3) lack of training programs and courses; 4) low priority given to instructional computing; 5) lack of incentives for teachers; and 6) the need for greater administrative commitment and recognition. What is required is a multi-level approach for dealing with the entire set of problems. Each will be discussed in turn.

The need to certify teachers of computer science is pervasive. Currently, only four states in the United States certify teachers of computer science. To illustrate the critical nature of this need, consider the following situation in a large school system which requires mathematics teachers to take 24 hours of mathematics content; yet, to teach computer science, the state requires no coursework in that area. The problem is not limited to computer science courses. Teachers are not required to take computer education courses in order to teach courses that involve computing in an adjunctive manner (e.g., computer applications in social sciences). Consequently, legitimate criticisms of computer misuse and of ill-prepared or unqualified teachers of computing exist.

Computer systems and instructional materials are proliferating at an amazing rate. Yet, it is surprising how many teachers and administrators are unaware of what the systems can do and what instructional materials exist. It follows that too many of them do not know how to effectively use computer-related materials.

Generally, in relation to other disciplines, university-based programs, courses, and workshops in computer education for teachers are lacking. Where courses do exist, they are often optional and not well-integrated in degree programs. (A few programs in computers and education do exist, for example, at Teachers College, the University of Massachusetts and the University of Illinois). In all fairness, though, the demand for such training is not being felt by universities.

Many educators do not seem to see the revelance of computing in the curriculum, and, therefore, it is given a low priority. High school teachers, in particular, sometimes complain that there is no room for computing in existing curricula. But, computer-related materials are being incorporated into textbooks and other curriculum materials signalling a philosophical advocacy for computing by curriculum developers. If Minsky (3) is correct in his prophetic statement that "eventually, programming itself will become more important even than mathematics in early education", then we must deal with teacher training to a greater extent than is presently the case.

Because computer science is a new discipline to many teachers, there is a seeming reluctance to pursue the study of it. It involves both new ways of thinking and new approaches to pedagogy.

Paradoxically, teachers seem to recognize the need for training in computing. In a recent survey conducted by the Minnesota Educational Computing Consortium (4), 85% of 1300 teachers surveyed agreed or strongly agreed that the secondary school student should have a minimal understanding of computers. Only 39% of the entire group agreed or strongly agreed that their training was adequate for using computers in instruction. When divided into users and non-users of computing, 62% of the users felt prepared. In any case, it appears that teachers, in general, perceive the need for preparation in computer use.

Administrative commitment to and recognition of instructional computing is sine qua non for effective and widespread use of computers in education. Administrators must be aware of the kinds of applications that exist and how these can be effectively used in education.

In spite of apparent lack of recognition of and commitment to instructional computing, some administrators do appear to recognize the need for more computer trained teachers. Dennis (5) surveyed 686 secondary school principals in Illinois and found that 1) 71% expressed a need for computer science teachers; 2) 55% expressed a need for state certification in computer science; and 3) 82% felt some computer science is valuable in the background of any teacher.

Baker (6) conducted a survey of 78 superintendendents in 50 states regarding their preferences concerning the roles of computer science and teacher training in secondary schools. Respondents stated their preferences with respect to: 1) a computer science department in secondary schools; 2) every teacher required to take three computer science courses; 3) a computer science department and every teacher trained in basic uses of the computer; and 4) none preferred. When divided into users and non-users of computing, a majority of the users preferred option 3, and the majority of the non-users preferred option 1 or 3.

Generally, administrators seem to have mixed feelings about computer education for teachers. Perhaps a comprehensive national survey would provide data on administrators' attitudes for the purpose of planning and decision making.

In spite of these problems and inhibiting factors, we do have available alternatives that could serve as prototypes for educating teachers. Existing and previous approaches for educating teachers in the instructional uses of computers are summarized in a volume edited by me (1). A few university-based programs exist and range from B.S. degree to summer M.S. degree programs. In addition, non-profit organizations and consortia such as the Minnesota Educational Computing Consortium, the Oregon Council for Computer Education, and Project LOCAL have been very effective in disseminating ideas and materials to schools. Finally, federally funded programs such as NSF's Programs in Pre-College Teacher Development and Information Dissemination in Science Education and Title IV-C programs of the U.S. Office of Education offer still other ways to educate teachers.

A Framework for Training Needs

How much should a teacher know about computer use in education? This is a complex question for which no simple answer exists. We can deal with this question based on what is taught, what certification and experience requirements exist, and what is believed necessary by experts. The matter of certification must be dealt with by school decision makers. The other matters are dealt with through the framework presented in this section.

In order to facilitate planning for and discussion of teacher education in computing, the general framework presented in Figure 1 is given. It deals with the kinds of courses and experiences that are necessary for various types of computer-related teaching. Its conceptualization is based on this author's experiences in providing computer education courses to a variety of teachers over the last nine years, as well as recommendations for courses by the Association for Computing Machinery (7).

COURSES/EXPERIENCE

TYPES OF TEACHING	Instructional Design	Designing CBLM	Programming	Hardware & Software Organization	Computer Uses In Education	Computers & Society
Computer-assisted Instruction	X	X	X		X	
Computer Science, Data Processing			X	X		X
Computers and Society			X	X		X

Figure 1. A Framework for Training Needs.

Instructional Design: An Introduction

This course is designed to provide a basis for understanding and applying principles of instructional technology in education. It includes: 1) an overview of instructional technology; 2) aspects of instructional planning, development, implementation, and evaluation (e.g., task analysis, behavioral objectives, presentation strategies); and 3) selected means for individualizing learning (e.g., CAI).

Designing Computer-Based Learning Materials (CBLM)

Depending on background and interests, the student learns how to design, implement, and evaluate instructional materials using computers. The primary performance products are computer-based instructional or management modules. The student demonstrates mastery of selected computer programming techniques germane to project development. The content includes: 1) computer programming for instruction; 2) review of prototype computer-based materials; 3) development of instructional or management modules; 4) implementation; and 5) evaluation.

Programming

In a first course, Computer Programming I, the content recommended by the Association for Computing Machinery (ACM) includes the development of algorithms and the design, coding, debugging, and documentation of programs written in widely-used, high-level programming languages. Computer organization is also introduced.

In the second course, Computer Programming II, students are introduced to structured programming concepts, string processing, searching and sorting, data structures, and recursion. Analysis of algorithms is discussed.

These two courses or some equivalent would appear to be relevant to teachers who either major or minor in computer science or who engage in the other types of teaching described in Figure I.

Hardware and Software Organization

The ACM has recommended and described other courses in computer science that would follow the two programming courses. These courses involve organizational and design aspects of computer hardware and software. Courses recommended for a major in computer science are: assembly language programming, introduction to file processing, and introduction to computer organization. More advanced courses in operating systems and computer architecture, data structures, and programming languages are also suggested. When planning specifically for teachers, some subset of these may need to be selected especially for those who minor in computer science. Frederick (8) has provided such an example.

Computer Uses in Education

This course provides a basis for developing, implementing, and evaluating computer uses in education. Following are topics of such a course: 1) computer systems fundamentals; 2) instructional usage modes; 3) considerations in developing an evaluating computer-based instruction; 4) representative projects; 5) issues and problems. In such a course, students get hands on experience in various applications and demonstrate competence through some form of instructional evaluation or development effort.

Computers and Society

A course in computers and society for teachers should deal with systems fundamentals, programming, and applications in a wide variety of areas and their implications. Many excellent textbooks are available for use in such a course, such as Rothman and Mosmann (9) and Sanders (10).

Implementing Change

In order to expedite computer education for teachers, school decision makers must 1) identify needs; 2) establish goals; 3) develop certification requirements; and 4) implement teacher education and support options.

The framework discussed in the previous section could facilitate the identification of generic training needs, the establishment of goals, and the development of requirements for teaching computer-related courses by teachers. Although comprehensive national, state, and local action is required, specific actions could be implemented immediately. For example, all pre-service teachers could be required to take a course in computer programming even if it is offered in lieu of another content or discipline-related course.

Various types of teacher training and support activities are possible. Some of these were implemented by schools using computers that were identified as exemplary in a recent study by Hargan and Hunter (11). The activities included:

1) Consulting by professional staff or outside experts;
2) Department meetings and demonstrations;
3) Workshops offered for in-service teachers who may not wish to commit themselves to entire courses (Appendix A includes outlines for a set of workshops proposed by this author in collaboration with M. and J. Eisenberg in microcomputer usage. Because microcomputers are being introduced in schools on a steadily increasing basis, workshops may be an expeditious way to train teachers.);
4) Summer curriculum development teams for teachers with previous development experience;
5) Paid released time;
6) Staff development within school systems;
7) Memberships paid in professional organizations;
8) Travel paid to professional meetings;
9) University classes.

Finally, improvement in the existing state of teacher education in computer use requires the resolution of all the problems and issues identified in this paper. A framework for thinking about and planning for teacher education has been presented and suggestions for the implementation of change have been made. Widespread and effective use of computers in education requires such action.

REFERENCES

1. Milner, S. (ed.) Topics in Instructional Computing (Volume I); Teacher Education. New York: Association for Computing Machinery, 1975.
2. International Federation for Information Processing Technical Committee on Education (TC-3). Computer Education for Teachers in Secondary Schools: Aims and Objectives in Teacher Training. IFIP, 1972.
3. Minsky, M. Form and Content in Computer Science. JACM, 17,2, 1970.
4. Minnesota Educational Computing Consortium. Computer Literacy Study - an Update. St. Paul, MN: MECC, June 2, 1978.
5. Dennis, J.R., et al. Computer Activities in Illinois Secondary Schools. Urbana, Illinois: Department of Secondary Education, University of Illinois, June, 1977.
6. Baker, J. Computers in the Curriculum. Phi Delta Kappa Fastback No. 82. Bloomington, Indiana: Phi Delta Kappa, 1976.
7. Austing, R.H., Barnes, B.H., Bonnette, D.T., Engel, G.L., and Stokes, G., Editors. Curriculum '78: Recommendations for the Undergraduate Program in Computer Science — A Report of the ACM Curriculum Committee on Computer Science. CACM, 22, 3, 1979.
8. Frederick, T. Computer Science Education For Students Training To Be Teachers. In S.D. Milner (ed.), Topics in Instructional Computing (Volume I): Teacher Education. New York: ACM, 1975.
9. Rothman, S. and Mosmann, C. Computers and Society. (2nd. ed.). Chicago: SRA, 1976.
10. Sanders, D. Computers in Society. (2nd. ed.). New York: McGraw-Hill, 1977.
11. Hargan, C. and Hunter, B. Instructional Computing: Ten Case Studies. Alexandria, Virginia: Human Resources Research Organization, 1978.

Appendix A: Proposed Workshops

Workshop 1. Microcomputer Fundamentals
- A. Integrated circuits
- B. Microcomputer components
- C. Distinctions (maxi-vs midi-vs mini-vs microcomputers)
- D. Representation of data
- E. Software

Workshop 2. BASIC Programming I
- A. Programming fundamentals
- B. Introduction to BASIC
- C. Numeric applications
- D. Non-numeric applications

Workshop 3. BASIC Programming II
- A. Advanced numeric applications
- B. Advanced non-numeric applications
- C. File manipulation

Workshop 4. Machine and Assembly Language Programming
- A. Fundamentals
- B. Instruction sets
- C. Addressing

Workshop 5. Microcomputer Software
- A. Languages
- B. Transportability

Workshop 6. Microcomputer Peripherals
- A. Available peripherals
- B. Using available hardware components (e.g., printers, audio devices, floppy disks)

Workshop 7. Computer hardware experiments
- A. Digital logic experiments
- B. Counting (Binary, BCD)
- C. Clock usage

Workshop 8. Text Editing
- A. Fundamental concepts
- B. Editing practice

Workshop 9. Microcomputers and Society
- A. What is a microcomputer
- B. Applications
- C. Implications

STAGES OF DEVELOPMENT IN INTRODUCING COMPUTING TO TEACHERS

J. Richard Dennis
Department of Secondary Education
University of Illinois

ABSTRACT

For the practicing teacher, becoming proficient in the classroom use of computers requires special consideration of the work context and of the constraints on personal time and energy. Unlike the preservice counterpart, the practicing teacher does not possess opportunities for high intensity, lengthy learning experiences.

The process of becoming a computer-using teacher can be seen in three rather distinct stages of personal growth. Each of these stages has its own information, training, and resource needs. For the teacher educator and/or implementation facilitator, an understanding and concern for these stages can ensure more lasting effects from training efforts.

* * *

Although studies both nationally and regionally have shown steady growth of instructional use of computers since in the 1960s, (Bukoski and Korotkin, 1976; Dennis, Dillhunt, and Muiznieks, 1977) these studies also show that such activity involves a very small portion of the teachers in any school. Also, there is little evidence of the spreading influence of computing to the spectrum of subjects found in a typical secondary school. One of the major reasons cited for such narrow impact of instructional computing is the lack of teachers prepared for using computers.

The thoughts presented here are concerned with teaching with computers, although the comments also apply to those who would teach about computers.

The training models discussed have been developed and implemented in teacher education programs of the Department of Secondary Education, University of Illinois, Urbana. At this writing the models are being considered for implementation at three other teacher education institutions.

There are two conventional avenues of training for teachers: preservice and inservice. Training at each of these career levels has distinct characteristics. During preservice, an individual can enter into relatively intensive and lengthy learning experiences. Learning skills are at their sharpest and, due to minimal other distractions in the learner's life, attentions usually are focused intently. But preservice teachers have little experience on which to act or to judge their learning. Consequently, training must be structured to reflect the view of the practitioner. At the same time, preservice personnel are being prepared for the future. Particularly when training is in an area involving rapidly advancing technique or technology, prospective teachers require a foundation from which to act.

The inservice career differs considerably. Practicing teachers seldom enjoy long periods in which to intensively pursue a new topic. Personal learning skills are not as sharp, and attentions are fragmented. During the course of a school day a secondary teacher must attempt to deal productively and personally with 125-150 different people. During this time they must shift gears cognitively several times on only a few moments notice between class periods. Where the previous 50 minutes may have been spent in diagnostic and remedial teaching of basic arithmetic skills, the teacher may have to shift next to a lesson in pre-calculus for very talented students.

Clearly, inservice teachers have their attention on practice, and only where time will permit on the improvement of same. But the latter is a luxury. New plans for altering or improving instruction must be capsulated in such a way that concepts may be grasped and skills acquired from relatively short and infrequent investments of time. The only exceptions to this are summer or sabbatical training. Career teachers, however, frequently have other demands on their summers and the tightening of school budgets recently experienced (and expected) has made a sabbatical leave a rarity in public schooling.

A Model For Inservice Training In IAC

For the practicing teacher, training for classroom use of computers requires special consideration of the job context and the constraints on personal time and energy already discussed. With such constraints in mind we shall describe three stages of inservice training designed to bring a practicing teacher to a level of competence in teaching with computers. These stages are:

1. Awareness Stage
2. Implementing Stage
3. Maintenance of Growth Stage

Each of these stages has unique information needs and specific ability goals to be achieved.

At the Awareness stage, two types of teachers are encountered. First, there are teachers who teach in schools already having instructional computing resources, but who are not personally involved. The second type teacher encountered is the one from a school not yet having instructional computers. Both of these teachers are very much interested in the question:

> What could I do with an instructional computer?

For the teacher from an unequipped school, there is an almost equally important question to address at the awareness stage:

> How would my school get an instructional computer if desired?

For each of these types specific ideas are needed on how a computer might support or facilitate instruction in their subject. But one must keep in mind that seldom do these teachers have much background with computers or even the working vocabulary. So, relevant information must be framed in the context of day-to-day instruction with very little technical reference to the computer. Generally, such an information program is the subject of a session lasting a few hours. An essential attribute of such a program is the opportunity to view and to experience firsthand a variety of instructional computing activities. Such examples should be chosen from materials that have had actual classroom use, and wherever possible should include some description or other information on how the computer activity is integrated with (and benefits) regular instruction. Also at this stage, teachers receive their first introduction to the characteristics of the various instructional computer styles that influence their quality, effectiveness, or motivating power.

For the teacher not currently in a computer-equipped situation, an awareness program also should include a discussion of methods others have used successfully to bring computers into a school. For this teacher it is almost mandatory that a responsible administrator be a co-participant in the awareness activities. Both the teacher and the administrator should be convinced of the need to define as explicitly as possible the uses they will make of a computer before attempting to purchase major amounts of equipment. In the absence of such careful procedure, the risks are great of acquiring equipment in some way incompatible with or inadequate for uses likely to emerge later.

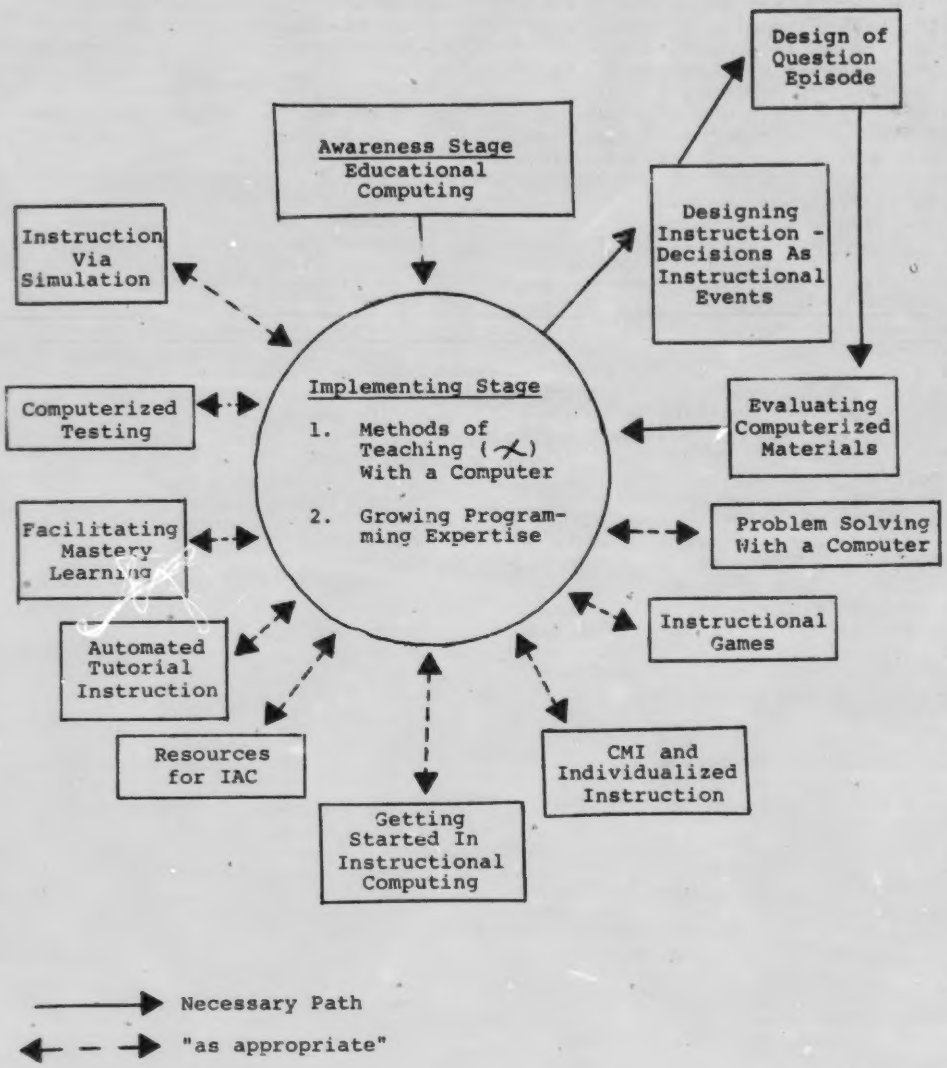
Inservice teachers at the Implementing stage of instructional computing require two kinds of training, both of which imply rather intensive commitments for a period of time. One of these is a detailed look at instructional computing methods and their characteristics. Integrated with this training should be a growing expertise with instructional programming. The inservice teacher is much more action oriented than the preservice counterpart. In most cases, this precludes a succession of experiences from programming instruction to intensive looks at computerized instruction. For the inservice person, we recommend the integration and simultaneous growth of these abilities so that classroom implementation (and experiences) may begin very early in the training period.

For gaining a background in computing, one productive technique is to progress from reproducing instructional programs created elsewhere, to creating original instances oneself. In this way the teacher can learn rudiments of a programming language (commands, formats, etc.) while quickly achieving an activity usable in the classroom. This postpones until later the more difficult tasks of developing program algorithms. In the meantime, valuable classroom experiences with computers can be derived in the instructional environment. These piloting experiences help to rapidly build perceptions of those applications of a computer likely to be of maximum personal use.

The teacher has a particularly difficult problem in providing appropriate resources during this implementing stage. This is most easily solved in cases where the teacher training can be exported to the audience's school and computing equipment is already there for use. Where either of these conditions is not present, the teacher must take steps to insure that efforts expended on pilot implementations can be transferred easily to the eventual equipment and software configuration installed at the schooling site.

Another often overlooked demand is that resources be readily available to the trainee. All facilities must be made as easy to access and use as possible. Example programs must be ready and must work. Documentation must be available and must be accurate. In part, such demands serve as a model to the trainees. More important, however, is the need to eliminate or avoid as many levels of frustration for the trainees as possible. We already have said that for the practicing teacher, time is at a premium. Any nonproductive use of this resource has significant deterring effects on the audience.

A successful curriculum map for inservice training at the implementation stage is shown in Figure 1. Note that the plan reflects the central theme of a subject characteristic of the


Design of Question Episode
Awareness Stage
Educational Computing
Instruction Via Simulation
Designing Instruction - Decisions As Instructional Events
Implementing Stage
1. Methods of Teaching (x) With a Computer
2. Growing Programming Expertise
Computerized Testing
Evaluating Computerized Materials
Facilitating Mastery Learning
Problem Solving With a Computer
Instructional Games
Automated Tutorial Instruction
Resources for IAC
CMI and Individualized Instruction
Getting Started In Instructional Computing
Necessary Path
"as appropriate"

CURRICULUM MAP FOR
IAC TRAINING
(IN-SERVICE)

FIGURE 1

orientation of an experienced teacher. Integrated with this focus are the detailed studies of the various instructional computing styles as they are deemed appropriate to the particular teacher and subject. Note also that this plan includes provision for informational resources to those implementing teachers also involved in initial computer resource procurement. These teachers will require a wide variety of information about hardware, software, and applications. They usually will need extensive consultation and assistance in relating desired computing tasks to equipment needs.

Many attempts to install new programs or activities into secondary school practice have failed during the implementing stage. During the 1950s and 1960s many attempts at installing new curricula or new instructional models in the United States were made. Many people were trained to use the various new curricula. They went away from training apparently highly motivated to proceed. But a short time later it was not uncommon to find a lapse to more conventional curricula or instructional models. Sustaining the efforts was another matter indeed.

The same phenomenon is observable in some instances of instructional computing installation. Resources are made available, teachers receive initial training, perhaps to the extent of having an initial set of computer activities prepared, but some time later the teachers have not progressed into expanding their repertoire. And not infrequently, initial efforts have ceased altogether.

Sustaining implementation efforts in education is not a new problem. Some would argue that this is neither the responsibility nor the concern of the trainer. This author cannot accept such a view. Instead, the implementing stage must be structured to include a sustained new information flow, readily available consultation on problems surrounding implementation, and a great deal of hand-holding. Periodic debriefings on past experiences are in order. Consultations of an advisory or confirming nature are needed to reinforce correct application of past training.

Practicing teachers probably should not attempt major implementations in isolation. Several teachers attempting similar implementations can provide each other with a level of support and consultation difficult to achieve from outside the school. Added to this are needed periodic third party debriefing, confirmation, and future efforts sessions to sustain momentum, to confirm perceptions, and to reconcile observed weaknesses in recent efforts.

To date, only a few teachers at the pre-college level have progressed to what we are calling the maintenance or growth stage of instructional computing expertise. It is at this stage that the teacher ceases somewhat incidental computer instruction activities or experiments and, instead, establishes a major role for the computer in the classroom environment. That role may range widely depending upon the teaching strengths or interests of the teacher, the nature of the subject, the teacher's view of the objectives of the subject, etc. One teacher oriented toward individualized instruction may set about the creation and implementation of an elaborate instructional management facility. Another teacher, wishing to emphasize a minimal level of performance in processes or procedures of the subject, may design and implement several simulations. A department, wishing to improve and expedite its testing program, may set about to create a computerized test-item banking/test construction system. A group of foreign language teachers, wishing to inject another dimension of culture education into their program, may elect to construct simulations of trips to foreign lands--each decision in such a simulation providing information about economic, religious, social, or political characteristics of the country and the implications of same for the foreign traveler. The possibilities for intensive or group projects at this stage are boundless.

The role of the trainer changes radically at this stage. Where before, the trainer had an initiating responsibility, now the educator need only assume an on-call posture. The computer-using teacher has come of age. The on-call consultations will serve largely to prevent reinvention of wheels. These wheels normally take the form of computer software systems available elsewhere and tailored to achieving certain instructional tasks. The trainer is in a better position to survey available materials. The teacher still may solicite evaluative opinions, but the motivation for such solicitations clearly is more courtesy or flattery than uncertainty.

From this point on, the computer-using teacher is self-perpetuating. The trainer has achieved a goal.

Implementing The Inservice Model

Institutions normally involved in teacher education regularly have engaged in dissemination of knowledge about new programs or instructional models. For such institutions to engage in continuing education programs such as that described above requires major changes in staffing patterns and acceptable faculty activities. Such changes are not impossible. But because of classical rank and pay evaluation procedures, the role, for example, of a facilitating consultant and evaluator is a difficult one to reconcile to higher education administrations.

One way to bootleg such input to small numbers of schools is to organize pre-service teacher education and educational research efforts in the same schools where instructional computer implementation is taking place. The research component of such a trilogy provides a justification for major levels of contact and effort by the trainer. Of course the scale of such efforts is limited.

That such services are valuable to schools or teachers attempting major implementations partly is attested to by the growth and popularity of teacher centers over the past several years. One of the major functions such centers provide is an opportunity for interaction among teachers either looking for ideas or looking for consultation on personal implementation efforts. Without sustaining input during the awareness and implementation stages of teacher change, new ideas for schooling rapidly fade and die in their infancy.

Bibliography

Bukoski, William and Korotkin, Arthur "Computing Activities in Secondary Education", Educational Technology, Vol. XVI, No. 1, , January, 1976, (pp. 9-23).

Dennis, J. Richard, Dillhunt, Clifford and Muiznieks, Viktors J., "Computer Activities in Illinois Schools", The Illinois Series on Educational Application of Computers, Department of Secondary Education, University of Illinois, Urbana, Illinois, Number 24, June, 1977.

COMPUTING COMPETENCIES FOR SCHOOL TEACHERS: A PRELIMINARY PROJECTION FOR THE TEACHER OF COMPUTING

*James L. Poirot, North Texas State University,
Denton, Texas, (817) 788-2767
John Hamblen, University of Missouri — Rolla
James D. Powell, North Carolina State University
Robert P. Taylor, Teachers College, Columbia University

Introduction

This paper is one of several dealing with the development of curricular and teacher training guidelines for the intergration of computing into the elementary and secondary schools of the country. It represents part of the continuing work of the Elementary and Secondary Schools Subcommittee (ES^3) of the ACM Curriculum Committee. Only matters related to presentation of the teacher of computing science as a separate subject in the school are discussed. The computing education of all other teachers is treated in "Computing Competencies for School Teachers -- A Preliminary Projection for all but the Teacher of Computing" by the same authors (Taylor, et al 1979), a paper which also contains a more complete introduction than this one.

Many papers have been written on the general problem of computing education for school teachers; those papers and the ACM Curriculum '78 guidelines have been freely drawn upon in drafting this paper. While many of those papers suggest specific courses and programs, this paper focuses upon competencies. Rather than approaching the problem of how to educate teachers of computer science, it attempts to specify the competencies needed by a teacher. Once competencies are defined, training programs with a variety of approaches can be designed.

The Teacher of Computing in the Schools--General

The teacher of computer science at the pre-college level should be able not only to teach courses in computer literacy, computer programming, and computer applications, but also be a resource for the gifted child pursuing advanced projects, fellow teachers wishing to use the computer as a tool, and the principal of the school needing help in administrative applications. Competency in computer programming alone is not sufficient for the teacher of computing.

Since much of the knowledge required for a computer science teacher is similar to that required of a computer professional, many of the computer competencies defined in the recent ACM Curriculum Committee report Curriculum '78 apply to such a teacher. Therefore, this report was used extensively in arriving at the recommended teacher competencies. A very important assumption that we make is that the teacher of computer science is exactly that, a teacher. Thus the competencies we list are related primarily to the area of computer science and computer education.

The task force realizes that the vast majority of educational programs have been implemented by dedicated teachers who have had little opportunity to obtain formal computer science education. We wish to stress that by listing desired competencies, we are not implying that these competencies must be gained by formal coursework. Many teachers already have acquired these skills by self-study and on-the-job training. This work should help and guide those individuals and institutions attempting to implement teacher education programs in computer science.

Teacher Competencies

The core material recommended for teachers of computer science represents esstential elementary material, as well as material especially designed for educators. Computer science teachers should:

1. be able in a reasonable amount of time to write programs that work correctly, are well documented, and are readable;
2. be able to determine whether or not they have written a reasonably efficient and well-organized program;
3. know what general types of problems are amenable to computer solution, and the various tools necessary for solving such problems;
4. understand basic computer architectures;
5. know adequate motivational arguments for teaching computer science topics at the pre-college level and know these course objectives;

6. know what educational tools can be uniquely employed in computer science education;
7. develop the ability to assist in the selection, acquisition, and use of computers, interactive terminals, and computer services which are suitable to the enhancement of instruction; and
8. be able to assist teachers in evaluation, selection, and development of appropriate instructional materials which use computing facilities.

The first four competencies are of the sort commonly needed by all computing professionals and are listed in Curriculum '78 as among those to be covered by the undergraduate computer science degree program. The last four competencies are not commonly needed by all computing professionals. They are essential only to computer science teachers.

Topics of Study to Attain Competencies

It should be stressed once again that the problem of competency attainment is not being addressed in this paper. Thus we do not list specific courses or programs of study. To clarify further the level of expertise to be expected of a computer science teacher, the following topics should be part of all teacher preparation programs:

Programming Topics: Includes algorithms, programming languages, programming style, debugging and verification, applications.

Software Organization: Includes computer structure and machine language, data representation, symbolic coding and assembly systems, addressing techniques, macros, program segmentation and linkage, linkers and loaders, systems and utility programs.

Hardware Organization: Includes computer systems organization, logic design, data representation and transfer, digital arithmetic, digital storage and accessing control, I/O reliability.

Data Structures and File Processing: Includes data structures, sorting and searching, trees, file terminology, sequential access, random access, file I/O.

Teaching Computer Science: Includes techniques of teaching, motivational whys of teaching and objectives to be reached when teaching computer topics, using computer tools such as games and simulations.

Computers in Education: Includes computer-assisted instruction, computer-managed instruction, social implications of computers in education, administrative applications.

Once again we rely on Curriculum '78, along with a vast amount of research in computer education, to support the inclusion of Programming Topics, Software Organization, Hardware Organization, and Data Structures and File Processing.

We feel the topics of Teaching Computer Science and Computers in Education are essential to any program of teacher preparation. Materials on why and how to teach computer science are invaluable to a teacher. Most research conducted on first courses in computing has concluded that computer programming alone is not sufficient. Thus other computer-related material in a beginning course should be included as part of teacher preparation program.

Including study of Computers in Education will allow the teachers to help provide direction to school systems that are integrating computing into their curricula. This additional background should allow the computer science teacher to assist in fostering the development and implementation of computing throughout the school.

Future Steps

Having guidelines for teacher competencies allows us to now define a much greater (and more pressing) problem: development of adequate teacher education programs. This problem may be broken into two parts, preservice and inservice teacher education.

Preservice programs involve educating future teachers in standard educational programs. Certification programs for the computing teacher [7], have already been implemented in several states, but much work still remains. For example, results of survey conducted by Justine Barker during the 1975-76 school year and published in Computers in Education showed that only 6.8% of the 1975 major universities surveyed offered a complete curriculum for teachers. The competencies finally worked out for the computing teacher should be translated into training for this teacher of a new subject, in teacher training institutions across the country.

Inservice teacher education is required for teachers already in the field. The need for well-defined programs in this area is urgent, since for economic and other reasons, many teachers across the country are being pressed into teaching computing science even though they have no formal training in the area. The competencies worked out for the computing teacher must be translated into other programs for these people, programs that are scheduled so that working teachers can take them.

Clearly, much remains to be done. It will require computer scientists to work closely with education faculties designing programs that will teach competence in both computing and education. The task force will continue to work on defining competencies and identifying broad program outlines, but we welcome help from colleagues for our subtask or for other aspects of the work of ES^3 as a whole. We believe the need demands the effort.

Bibliography

1. Atchinson, William F. "Computer science preparation for secondary school teachers". *SIGCSE Bulletin*, 5, 1:45-47 (1973).
2. *Conference On Basic Mathematical Skills and Learning*, U.S. Department of Health, Education and Welfare. Euclid, Ohio, 1975.
3. *Curriculum '78: Recommendations for the Undergraduate Program in Computer Science*, A Report of the ACM Curriculum Committee on Computer Science, December, 1978.
4. Danver, Jean H. "Suggestions for programs". Hanover, N.H.: Keiwit Computational Center, Dartmouth College, 1970.
5. Dennis, J. Richard, Dillhung, C., and Muiznieks, J. "Computer Activities in Secondary Schools in Illinois", *Illinois Series on Educational Applications of Computers*, No. 24, Univ. of Illinois, 1977.
6. Esterson, D.M. "Problems of Implementation: Courses for Pre- and In-Service Education". *Information and Mathematics In Secondary Schools*. North-Holland Publishing Co., 1978.
7. Moulton, Peter and Moursund, David. "A summer Master's degree program in computer science". *Topics in Instructional Computing SIGCSE*. Vol. 1, 31-36, January, 1975.
8. Poirot, James L. "A course description for teacher education in computer science ". *SIGCSE Bulletin*, Vol. 8, 39-48, February, 1976.
9. Poirot, J.L. and Groves, D.N. *Beginning Computer Science*. Sterling Swift Publishing, Manchaca, Texas, 1978.
10. Poirot, J.L. and Groves, D.N. *Computer Science for the Teacher*. Sterling Swift Publishing, Manchaca, Texas, 1976
11. Poirot, J.L. and Early, G.G. "Teacher certification -- a computer science necessity". *Topics in Instructional Computing SIGCSE*, Vol. 1, 15-18, January, 1975.
12. *Recommendations Regarding Computers in High School Education*, Conference Board of the Mathematical Sciences, Washington, D.C., April 1972.
13. Spencer, Donald. "Curriculum and Teacher Training for Secondary Schools", *Interface*, Vol. 5, No. 2, April, 1971.
14. Statz, Joyce. "Training secondary school computer science educators". *Topics in Instructional Computing SIGCSE*, Vol. 1, 4-9, January, 1975.
15. Taylor, Robert P. "Graduate Remedial Training in Computing for Educators", *SIGCSE Technical Symposium Proceedings*, Dayton, February, 1979.

COMPUTING COMPETENCIES FOR SCHOOL TEACHERS:

A PRELIMINARY PROJECTION FOR ALL BUT THE TEACHER OF COMPUTING

Robert P. Taylor, Teachers College, Columbia University
James L Poirot, North Texas State University
James D. Powell, North Carolina State University
John Hamblen, University of Missouri-Rolla

Introduction

This paper reports the preliminary thinking of a task-group of the ACM Curriculum Committee's Elementary and Secondary Schools Subcommittee. It assumes that computing must be integrated into the training of all teachers and outlines the extent of this integration by specifying competencies which that training should generate.

Computing competencies needed by teachers

There are three sets of computing competencies which must be defined for the integration of computing into the schools. The first includes those which *all* teachers must have, regardless of their level or discipline, even if that discipline is the teaching of computing itself. That set is dealt with in the first main subdivision below: "1.0 Computing competencies needed by all teachers." The second set includes those needed only by the teacher of computing as a subject. That set will not be discussed in this paper, but in a separate one [1]. It should be noted however, that that second set presupposes and builds upon the set specified in 1.0 below. The third set includes additional competencies for teachers who use computing to support or enhance instruction in subjects other than computing. (One may apply primarily to the art teacher; another to the reading teacher; a third to the physics teacher; and so on.) Any given teacher will require at least one of the competencies included in the third set but no one of them will be universally appropriate for all teachers. This third set is discussed in the second major subdivision below: "3.0 Subject-specific competencies needed by teachers."

1.0 : Universal computing competencies needed by all teachers

There is a set of computing competencies which all school teachers should have henceforth if they are to improve their work by using computers. These universally required competencies relate to either or both of two goals: (1) to *understand* computing and (2) to *use* computing. They can be stated partially in terms of competencies listed in ACM's "Curriculum '78" and partially in terms of different competencies, derived from other sources. A number of such other sources are listed in the references at the end of this paper. Those sources reflect the abundance of diverse work that has taken place in the past decade, relating computing to education.

In terms of these universal competencies, *every* teacher should:

C1.1 be able to write programs and subprograms that work correctly and are easily readable by others, and be familiar with how such programs and subprograms fit together into systems;
C1.2 know what general types of problems are amenable to computer solution and the various tools necessary for solving such problems, particularly in using computers in education;
C1.3 know what general types of problems are *not* currently amenable to computer solution and why not, particularly in using computers in education;
C1.4 be able to discuss at the level of an intelligent layperson the history of computing, particularly as it relates to education;
C1.5 be able to identify and generally rate several alternate sources of best current information on computing as it relates to education; and
C1.6 be able to discuss moral or human-impact issues of computing as they relate to: (a) societal use of computers generally, and (b) educational use particularly.

These competencies should be transmitted within the general preparation programs for *all* teachers by having those programs include the topics listed below (T1.1 through T1.4). For those being trained to teach computer science, these topics will represent only a small subset of what must be learned about computing and its uses [1]. For all other teachers however, apart from the subject-specific competencies covered below, these topics cover much of what must be learned about computing by the teacher who is to be minimally literate.

- T1.1 Programming Topics: Includes algorithms, programming languages (at least two), programming style, debugging and verification, task-specific programming for educational applications.
- T1.2 Data Structures and File Processing: Includes file terminology, sequential access, random access, file I/O, introduction to general data structures, introduction to sorting and searching.
- T1.3 Classic Applications of Computing in Education: Includes representative experience with problem solving and text manipulation, simulation, drill and practice, and complex tutorial systems including complete student progress record keeping and educational administrative systems.
- T1.4 Human/Machine Interfaces and Relationships: Includes experience with evaluating simple human/machine interfaces, acquaintance with contrasting points of view on such relationships and why they are held -- particularly the humanists' views of computing, and practice exploring the broader implications of human use of, and dependence upon, computers.

In T1.1, emphasis should be placed upon interrelationships among programs and subprograms, upon being able to call upon and use well-defined modules created by others. If time must be restricted, it would be more important to cover blocks or procedures (including simple and parametric, global and local) than to spend too much time on such topics as alternation and repetition control structures. An approaching featuring blocks has been used successfully without undue difficulty.[2] [3]

Most of the general teacher's work with computing back in the classroom will require a good basic understanding of how programs and routines are used and linked through parameters and calls; much less of that work will require the teacher to do detailed programming. The essence of readability and of top-down design concepts can be conveyed if blocks are understood thoroughly, even if the details of alternation and repetition are not. The reverse is not true. Besides, to pursue the topics in 3.0 below, the teacher will be well-served by a good understanding of how to use the work of others and of how various systems use parametrically linked modules. Whatever is covered here should be consistent with Curriculum '78, and should be designed to modularly prepare selected teachers to go on to the specialization required for teaching computing as a subject.

In T1.2, some knowledge of data structures and sorting and searching is probably important, but it is unclear whether much is essential. In any case, to pursue T1.3 with insight, the teacher should probably be given some minimal understanding of how to programmatically handle both graphic data and sonic data, two kinds that will be increasingly important in instructional applications. Whatever study of data structure concepts is essential to convey such understanding should be included. As with T1.1, whatever is covered here should be consistent with Curriculum '78.

T1.3 should certainly familiarize the teacher with several of the existing well-developed CAI systems cited in Section 3.0 below. Teachers will not develop new ideas about what could be done in their areas unless they see the best of what has been done; neither will they get a full understanding of what can not be reasonably well-done by computer without such exposure. For example, acquaintance with the physics system designed by Bork [4], with the CCC drill and practice systems based on Suppes's work [5], or with PLATO work [6] should serve to acquaint the teacher with the CAI issues.

Since many teachers end up in administration and since administrative uses of computers affect the teacher, some introduction to one or more representative administrative systems should be included. A student records system would probably be a reasonable choice for illustrative study -- it deals with information familiar to the teacher without using financial details some might find difficult to understand.

Teachers should also be familiar with word processing and super calculator modes of computer use as classic applications, though as home computers become more common perhaps little formal work will be necessary. Clearly word processing must be covered; every teacher does so much word processing manually that none should be left ignorant of how much word processing help the computer can give.

With respect to a combination of T1.1 and T1.4, the long range and the immediate implications of computing as a form of artificial intelligence should be taught. The excitement of learning to think about thinking and of contemplating the powers and limits of human intelligence are so significantly linked with computing experience that this aspect must be studied -- the opportunity is too great to pass up. Artificial intelligence may best convey both the power and limitation of computing in education. Acquaintance with any of several perspectives on this experience is essential and can be taught using such projects as the LOGO work [7] or the SOLO work [8].

3.0 : Subject-specific computing competencies needed by teachers

In addition to the set of universal competencies needed by all school teachers, there are additional level- and subject-specific competencies which teachers should have. The definitions of these competencies spring entirely from the vast and highly diverse body of experience with using computing in education over the last decade. Sources representing some of this work are listed in the bibliography. The competencies can be stated generally, irrespective of the teacher's eventual level or subject; the topics, though, will vary considerably, depending on both.

In terms of these subject-specific competencies, the teacher should:

C3.1 be able to use and evaluate the general capabilities of the computer as a tool for pursuing various discipline- or level-specific educational tasks;

C3.2 be able to use and evaluate alternative hardware and software systems designed to function as tutors or teacher aids; and

C3.3 be familiar with alternative hardware and software systems designed to perform school administration.

These competencies should be developed by the teacher preparation program, tailored to suit the trainee's intended teaching level and subject. We will not attempt to present an exhaustive list of topics corresponding to these subject-specific and level-specific competencies -- more work, more time, more knowledge, and far more help from others would be required first. Instead, we will present the topics for a few selected subjects and levels, to suggest how a full catalogue might be constructed.

Topics for Teachers of Primary Grades 1-4 -- TEC

TEC3.1 Computerless preparation for computing Experience with a wide range of computerless but computing-related activities that children can participate in to enhance their readiness to understand and work with computers.

TEC3.2 Games and Simulations Experience with a wide range of games and simulations that stimulate children to explore and better understand fundamental concepts and strategies of basic learning.

TEC3.3 Tutorial systems Experience with simple and complex tutorial systems focussing upon mathematics, spelling, reading, and other elementary topics, including bi-lingual variations of such systems; and

TEC3.4 Exploratory programming systems Experience with well-developed exploratory systems where child-appropriate I/O subsystems such as robots are programmatically manipulated by the child in a discovery or problem-solving approach.

Less work has taken place in the the area covered by TEC3.1 than one might expect. Despite the likely wide-spread availability of micro-processors in the immediate future, computerless computing activities can still be very useful. By contrast with heavily machine-dependent activities, they provide a more contemplative, less involved opportunity to examine some of the fundamental ideas connected with computing. They thus allow those using them to deepen their understanding even if they have access to computers. Typical examples may be found in some of Papert's work [9] and in Taylor [10].

Vast quantities of games and simulations are available for TEC3.2, but careful choices should be made in selecting them. Many are not well-written, either from a programming point of view or from a user point of view, and none should be selected unless it is both. Some of the best work in this area at this age level comes out of the People's Computer Company, under the initial stimulus of Albrecht [11].

With respect to TEC3.3, though much has been developed, not all of it is good. The experience of the teacher should certainly include at least one good system and some discussion of what lies behind it. The work of the CCC group under Suppes is certainly a worthy example in this category [12].

Finally, new exploratory systems relevant for TEC3.4 are appearing, but the pioneering work is still for illustration. In particular, the XEROX work which produced SMALLTALK [13] and the LOGO work, particularly as Seymour Papert has advertised and sustained it [14], is outstanding.

As microprocessor work evolves, where appropriate it should be built into this framework as further illustration for additional teacher experience breadth. Good examples already available are the Texas Instrument SPEAK-AND-SPELL micro and their DATAMAN micro. Both are limited but show what is coming.

Topics for Teachers of Foreign Language -- TFL

TFL3.1 Games and Simulations Experience with a wide range of games and simulations designed to provide cultural background and informal language learning, using the culture as context and the language as the medium of communication in the game or simulation.

TFL3.2 Tutorial Systems Experience with tutorial systems designed to enhance the learning of a foreign language through a carefully arranged body of interactive experiences driven by competency-based, computer-administered testing.

TFL3.3 Foreign Language Text Editing Experience with a powerful text editor used to create and manipulate texts in a foreign language

Under TFL3.1, simulations based on relevant activities and situations in the language-culture can provide insight into the language difficult to obtain in other ways. These, and many popular computer games, should also be used to provide a more informal language practice for learners. This practice can take two forms: translating and informally using the language. Teachers should become familiar with the practice of translating all user text of appropriate games and simulations into the target language, thus preparing for having their students do such translation. Teachers should also have wide experience with playing games whose user text has been translated into the target foreign language and which expects all responses to be in that language. Such playing in a foreign language can be a valuable informal enhancement. Experience with a wide range of such games and simulations may also suggest new, more appropriate ones which the teacher can create or have others, including students, create. Some attempt to create such new material (or new variations of old material) should be part of every foreign language teacher's training. Naturally, as audio is available, it should be used.

There are many examples of language drill and practice suitable for use under TFL3.2. Work such as that done by Suppes [15] at Stanford should certainly be familiar to language teachers, though alternatives certainly exist.[16] Work in this area should rely as heavily upon audio and graphics as possible, thus cutting down on the automatic tendency to always translate from the native language and to develop competence only in reading and writing the foreign language.

TFL3.3 should ensure that teachers become comfortable with using a suitable computer text editor to manipulate text in the target foreign language. With appropriate accent mark capabilities, such an editor can encourage language learners to practice much more prose writing and thus enhance their overall command of the language.

Topics for Teachers of Physical Science -- TPS

TPS3.1 Exploratory Programming Systems Experience with well-defined exploratory systems through structured, discipline-appropriate languages including graphic I/O, programmatically manipulated by the student in discovery or problem-solving activities oriented to enhancing understanding of the physical sciences.

TPS3.2 Tutorial Systems Experience with tutorial systems designed to enhance learning of the physical sciences through a carefully arranged body of interactive experiences driven by competency-based, computer-administered testing.

TPS3.3 Games and Simulations Experience with stand-alone games and simulations designed to enhance understanding of specific physical phenomena or significant past scientific experiments.

The possibilities under all three topics for physical science teachers have been extensively explored already by Bork [17] and others.[18] Their careful work and well-documented analysis of it should be extended and incorporated in the training of physical science teachers. The training of such teachers should include experience with relevant examples which illustrate the three topics; it should also require each trainee to construct selected, similar small modules of computer-supported instruction as a normal part of teacher training.

Summary

This ES3 document is a preliminary projection of the computing-related competencies needed by elementary and secondary school teachers.

The entire projection is presented as a start, not a final document. It is intended to elicit criticism and reflective refinement from as many as care to thoughtfully give it.

Cited References

1) Poirot, James, James powell, John Hamblen, Robert Taylor. "Computing Competencies for School Teachers -- A Preliminary Projection for the

Teacher of Computing". National Educational Computing Conference Proceedings, Iowa City, 1979.

2) Levine, D. R. "The Simple Internal Procedure as an Effective Tool in the Teaching and Practice of Structured PL/1" SIGCSE Bulletin, vol 9, no 1 (Feb. 1977).

3) Taylor, R. P. "Teaching Programming without access to Computers" ICCA Proceedings (Conference on computer applications in developing countries sponsored by IFIPS and Canadian AID) Bangkok, 1977

4) Bork, Alfred "Learning to teach via teaching the computer to teach." Journal of Computer-based Instruction November 1975, vol 2.

5) Suppes, Patrick. "Computer-Assisted instruction at Stanford" in Man and Computer, Karger, 1970.

24) Smith, Stanley and Bruce Arne Sherwood "Educational Uses of the PLATO Computer System" SCIENCE vol 192 (Ap 1976)

7) Papert, Seymour. "Teaching Children Thinking", Logo Memo 2, MIT AI LAB, October, 1971.

8) Dwyer, Thomas. "The Art of Education: Blueprint for a Renaissance" Creative Computing, (Sept/Oct 1976)

9) Papert, Seymour. "Teaching Children to be Mathematicians VS. Teaching About Mathematics" Logo Memo 4, MIT AI LAI, July, 1971.

10) Taylor, R. P. "Computless Computing for Young Children or What to Do till the Computer Comes" in Proceeding of the 2nd World Conference on Computers in Education. North Holland: Amsterdam. 1975.

11) PCC (collective authorship) What to do after your hit return. People's Computer Company, Menlo Park, California, 1975.

12) Suppes, Patrick and Elizabeth Macken "Evaluation Studies of CCC Elementary School Curriculums 1971-1975" Computer Curriculum Corporation, Palo Alto, 1976.

13) Kay, Alan "A personal computer for children of all ages" in Proceedings of ACM National Conference Boston, August 1972.

14) Papert, Seymour. "Personal Computing and the Impact on Education" edited transcription of a talk delivered at the Gerard P. Weeg Memorial Conference, as printed in the proceedings, Computing in College and University: 1978 and Beyond, University of Iowa, 1978.

15) Suppes, Patrick, Robert Smith, Marian Beard "University-level Computer assisted Instruction at Stanford:1975." Instructional Science 6 (1977), Elsevier Scientific Publishing Company, Amsterdam.

16) Tanabe,Y et al "CAI in Language Education" in Proceedings of 2nd World Conference on Computers in Education, North Holland/Elsevier 1975.

17) Bork, Alfred. "Preparing Student-Computer Dialogs -- Advice to Teachers." PCDP U. C., Irvine, July, 1976.

18) Dwyer, Thomas A. "Some principles for the Human use of Computers in Education." International Journal of Man-Machine Studies 3, July, 1971.

Other References

19. Conference Board of the Mathematical Sciences Committee on Computer Education, Recommendations Regarding Computers in High School Education, CBMS, Washington, DC, 1972.

20. Curriculum Committee on Computer Science (C3S). "Curriculum 78: Recommendations for the Undergraduate Program in Computer Science." (Preliminary Draft, dated autumn 1978).

21. IFIP Technical Committee for Education, Working Group on Secondary School Education: Computer Education for Teachers in Secondary Schools: Aims and Objectives in Teacher Training AFIPS, Montvale, NJ 1972

22. Topics in Instructional Computing, (a special publication of SIGCUE), 1975

23. Dennis, J.Richard "Undergraduate Programs to Increase Instructional Computing in Schools" Proceedings of Ninth Conference on Computing in the Undergraduate Curriculum, East Lansing, Mich, 1977

24. Leuhrmann, Arthur "Reading, Writing, Arithmetic, and Computing" in Improving Instructional Productivity in Higher Education Educational Technology Publications, 1975

25. Moursund, David "Report of the ACM Teacher Certification Subcommittee" SIGCSE Bulletin vol 9, no 4, Dec 1977, pp-8-18

26. Molnar, Stephen "Computer Literacy: the next great crisis in American education" Oregon Computing Teacher vol 6,no 1,Sept 1978

27. "Computers and the Learning Society" report of hearings before USHR, October 1977, US Govt Printing Office, 1978

Computer-Managed Education:
Providing Quality Education on a Large Scale

Robert J. McGlinn
Department of Computer Science
Southeast Missouri State University
Cape Girardeau, Missouri 63701

Lowell A. Carmony
Mathematics Department
Southern Illinois University
Carbondale, Illinois 62901

INTRODUCTION

Most state universities have lower division courses that are taken by tremendous numbers of students, and these general studies classes can put a great burden on the instructors and departments involved. In 1976, observing the resources the mathematics department at Southern Illinois University at Carbondale was allocating to its intermediate algebra course, Professor David Kammler sought college funds so that the department could develop a computerized management system for the course. This paper describes a project undertaken to automate the routine tasks of constructing and grading quizzes and exams, and the recording and averaging of grades of the more than 2000 students who annually enroll in the department's intermediate algebra course, GSD 107. As we shall indicate below, the system is designed not to infringe upon the instructor in the classroom, but to liberate the instructor from the paperwork involved in teaching such a class.

Basically, the system operates as follows: Upon request a clerk logs onto a terminal and invokes a program, TESTGEN, to select the questions for a homework assignment, short quiz, hour exam, proficiency exam, or final exam. The clerk assembles the questions, then duplicates and distributes the test to the proper instructor. Typically, the students code their responses on optical scan sheets which are then processed with the results being sent automatically to the department's terminal. The clerk then uses a second program, SCANGRDE, to grade the tests and enter the results into the computerized gradebook. Via a third program, either POSTCMS or POSTOS, separate grade summary sheets can be produced for each section.

The software for the system is a series of interactive FORTRAN programs designed to operate in IBM's Conversational Monitor System (CMS). The software is designed so that the system is maintained daily by a secretary-clerk with no prior data processing experience.

TEST GENERATION

The test generation package has a data bank of 4500 multiple choice questions thoroughly covering the content of the course. Professors John Hooker, Michael Skalsky, and Larry Wimp jointly assembled the initial version of this data bank. The questions are broken into modules that correspond to subdivisions in the course syllabus. The questions are not identified serially, but are labeled by a system which permits easy additions or deletions of questions or modules. For example, the problem with identifier MIXTURE-II-21 is the 21st question in the second submodule of the module on mixture word problems.

The tests are not printed either on a line printer or at the terminal because in an introductory course we must have completely standard notation, such as

$$\sqrt{b^2 - 4ac} .$$

Thus, the program selects questions by listing their identifiers. The clerk looks up the identifiers in a card file and on the back side of the card the question and the 5 possible responses will be found. For example, the reverse side of MIXTURE-II-21 reads:

> What weight of water must be evaporated from 40 pounds of a 20% salt solution to produce a 50% solution?
> (a) 22 pounds (d) 18 pounds
> (b) 20 pounds (e) None of the above
> (c) 24 pounds

The clerk assembles the question cards four to a page and xeroxes them. The only typing needed is to number the questions. Once numbered a thermofax ditto master is produced from the xerox copy and the tests are then duplicated, collated, stapled, and distributed in the usual manner. The responsibility for initiating the test generation process belongs, of course, to the instructor. The instructor completes a user request form to choose either a system-designed exam (quiz, hour exam, or final exam) or a user-designed exam. The system-designed exams cover predetermined modules but the questions are, of course, randomly selected. In a user-designed exam the instructor specifies the modules or submodules and the number of questions to be selected from each.

After the instructors have submitted their forms, the clerk logs on to the terminal and executes the program TESTGEN. This interactive program prompts the clerk to enter the information necessary in constructing an appropriate test. The output of TESTGEN is the identifier of appropriate questions, which are then assembled and reproduced as described above. Actually, TESTGEN prints two lists (containing the same questions in different orders) so that two forms of the exam can be produced, thereby reducing the possibility of cheating.

COMPUTER GRADEBOOK

The routine and tedious task of bookkeeping grades is also handled by the system, but without violating the students' rights of privacy. The system provides for up to sixty homework, quiz, and hour exam scores, and one final exam score, per student. The instructor, of course, determines the number of quizzes, hour exams, and homework assignments that his or her section will have. The instructor also determines how many, if any, of the homework assignments, quizzes, and hour exams the program should drop when computing averages. The clerk also informs the program on a section by section basis how the instructor wants the homework, quiz, and exam averages weighted in determining the final course averages. All this flexibility is built into the program to insure that the software can adapt to the instructor, rather than forcing the instructor to adapt to the constraints of the software.

Since manually entering the gradebook at the beginning of each semester would be a very time-consuming task, this process has also been automated. At our request, Admissions and Records makes a file available to our terminal containing the name, the student identification number, and the section number of each student enrolled in GSD 107. By issuing the command '107book' the clerk creates the initial version of the gradebook. The programs also assign each student a unique code consisting of a single alphabetic character followed by two digits. The primary purpose of this student code is in the posting of grades which is explained below.

As is their wont, students continually withdraw, change sections, or enroll late. These changes are efficiently processed using the text editing facilities of CMS. The clerk affects these changes by typing the command

```
                   .enroll
```

and then following the prompting, types

```
                   .$drop
                   .$add
or                 .$switch
```

and then follows the instructions given by the computer. After all the changes have been processed, the clerk enters the command

```
                   .listbook
```

which causes a new class list to be printed. Secondly, the command

```
                   .loadtape
```

causes a backup copy of the gradebook to be stored on tape. In effect the EXEC LOADTAPE submits an OS job to the central computing system to update our tape with a current copy of the gradebook. It is obvious that a backup copy of the gradebook must always be available. The backup has only been needed once in three years, however, and that was during the development stage and was due to a programmer's error, not a clerk's error.

TEST GRADING

Experience has shown that more than one method is needed to input all the grades into the gradebook and so, several procedures have been developed.

1. SCANGRDE

The first procedure is a completely automated approach using optical scan sheets. This method is primarily used with hour exams and final exams. Each section is provided with a sufficient supply of optical scan sheets. The students code their names, identification numbers, section numbers, and responses on these sheets. The instructor returns the answer sheets along with a correctly coded answer key to the clerk who sees that they are picked up by the Student Affairs Research and Evaluation Center to be processed through their optical scanner. Twice daily a courier collects exams to be graded and returns graded exams.

The optical scanner reads the exams and outputs the results on a magnetic tape. The tape is transported to the central computer center where a FORTRAN program, OPIN, reads the tape and creates a file on our OS data set, USERS.DATA107. When the clerk logs onto the terminal, a message is automatically printed indicating which batches of exams have been loaded into our OS data set. If the message indicates that BATCH7 has been loaded into the data set, then the clerk completes the grading process by entering the command

```
                   .scangrde7
```

The FORTRAN program SCANGRDE automatically records the raw scores into the gradebook. Since the instructors usually return the scan sheets to the students, SCANGRDE also prints a duplicate of the batch being processed for use if the student challenges a grade. SCANGRDE also prints summary information for each section including the grades, the mean, the percentage that correctly responded to each question, and a grade distribution. This information is primarily intended for the instructor. Not only does it provide a quick summary of the exam, but also it pinpoints the areas that caused the most difficulties. If an instructor wishes to post these summaries he may do so by cutting off the names, leaving the students to be identified by their unique student codes.

2. TERMGRDE

For short quizzes the use of optical scan sheets has proved to be unsatisfactory. First, the use of scan sheets causes an unacceptable time delay for daily quizzes which lose their effectiveness if they are not returned at the next class period. Secondly, it is a waste of scan sheets when they are used to administer frequent short quizzes. On the other hand, simply inputing the grades into the gradebook one by one at the terminal does not allow for any meaningful summary of the quiz results or for any item analysis since the individual responses to the questions are not available. TERMGRDE is intended to circumvent these difficulties.

Each quiz that is to be graded using TERMGRDE must have the special heading shown below placed on it.

GSD107 NAME____________________

Section 1 ______
code

McGlinn(8:00) ☐☐☐☐☐☐☐☐☐☐
1 2 3 4 5 6 7 8 9 10
Answers

The students enter their names, codes, and responses in the appropriate locations in the heading. In order to double check their answers they also are encouraged to circle their responses below the questions. If the instructor wishes, the students can be required to show their work on the paper. However, only the responses entered in the heading are used in the automated grading procedure.

The clerk initiates the procedure that grades the quizzes and records the results in the gradebook by issuing the command

```
.termgrde
```

Following the prompts of that program, the clerk uses the CMS text editor to create a temporary response file. One record is created for each student containing the student's code and answers. After this file has been created the program automatically begins execution. A summary sheet similar to that produced by SCANGRDE is printed for each section.

Experience has shown that the clerk can process short quizzes for a thirty-five student section in about ten minutes. Roughly half this time is spent creating the response file and half spent executing the program and printing the summary sheet. For short quizzes, TERMGRDE has proved to be very successful.

3. LOADGRDE

The third way to enter grades in the gradebook is to load the raw scores by using the command

```
.loadgrde
```

Exam scores can be entered in this manner, but it is not recommended. LOADGRDE is used primarily for recording homework scores. Instructors are provided with record sheets to list homework results in a convenient manner. For efficiency the homework grades are not given to the clerk on a daily basis, but are held by the instructor until five assignments have been given and graded. As with the other interactive programs, LOADGRDE is so simple to execute that it is virtually impossible to make a mistake. The clerk simply responds to the questions and directions as they are presented by the program.

4. CHNGGRDE

Grade changes must be made for a variety of reasons. A student with a valid reason for missing an exam may receive permission from the instructor to take a make up exam. Or a student who incorrectly writes his student code may fail to be identified by SCANGRDE or TERMGRDE. Or the instructor may have reasons for altering a grade. Conceivably, a question could be ambiguous or could have no correct answer listed. In such a case several grade changes would need to be made. Experience has shown that such problems occur rarely and when they do occur the data bank is easily corrected.

Of course, at the end of a work day, after all grades have been entered by whatever means, the command

```
.loadtape
```

should be issued to update the backup tape copy of the gradebook.

GRADE POSTING

Students seem to appreciate the opportunity to see how they stand in a class during the course of the semester and so, we periodically provide a class roster to each instructor displaying the grades and an average for each student. Under the new privacy act it is illegal to post a student's grade by name or social security number, so we use the student code (one letter, two digits) to identify the student.

Exam scores are printed as percentages to make them more meaningful. That is, a raw score of 17 out of a possible 20, would print as an '85.' Final exams are printed on a two-hundred point basis, thus 36 out of 50 prints as a '144.' Remember, of course, that the instructor determines the cumulative weights of the quizzes, homework assignments, hour exams, and the final, as well as the number of each to be dropped before computing the average.

When the final posting is run an additional column titled 'COURSE GRADE' appears, but no entries appear in this column. This is done so that the instructor can use the sheet to post the final course grades. Note that the computer does not compute letter grades, but simply averages. The determination of the letter grades is entirely up to the instructor.

Preferably on Friday, after all the grades for the week have been entered into the computer, the clerk requests the postings by entering the command

```
.postos
```

The program asks for the date and then submits an OS job to the computer. Alternately, the command

```
.postcms
```

can be used. This command submits a program through CMS and should be used when only a few sections are to be posted.

CONCLUSIONS

The system described above has been in operation for three years at Southern Illinois University. It has proved to be a practical, reliable, and economical means of managing the tremendous amount of paperwork associated with the GSD 107 course. The current project is not, of course, a CAI project. Southern Illinois University will not, at any time in the near future, be able to provide terminals for the 2000 students who yearly take GSD 107. On the other hand, computer management of routine and tedious tasks is a realistic way of helping to provide quality mass education. As we have tried to indicate, the present system is designed to liberate the instructor from the less interesting chores that go along with teaching such a class so that more effort might be spent on actual instruction. From the students' point of view, the system assures a uniformity in exams across sections and yields periodic postings to check their progress.

Our system is semi-portable. The interactive FORTRAN programs run under IBM's Conversational Monitor System but could easily be modified to run on other time-sharing systems. Our data bank of questions is, of course, closely tied to our GSD 107 course. Other installations would need to prepare a data bank of questions for their own purposes. Note, however, that this need not be restricted to a math class. Any large introductory course from Anthropology to Zoology would be appropriate.

AN ON-LINE DIRECTORY OF CAI RESOURCES

Dr. Kathleen M. Swigger
Michael Maner
East Texas State University

Imagine, if you will, the following scene:

A group of visiting dignitaries, whom you are eagerly trying to impress, is standing in front of a CRT watching a series of CAI programs that you have assembled for demonstration purposes. One of the members in your party suddenly turns to you and asks if you have anything on ecology. You respond, "Yes, of course. Now let me think, what name did I call that program. Mmm. I think it's this one. No, that's not the right one. Maybe it's this one. Oh, I must have called the wrong library. Let's try it over here...Hey, don't go away, I think I've found it...."

Sound familiar? Of course it does. As an institution grows and expands its use of Computer-Based Education, it soon discovers that some type of method for storing and retrieving information about CAI programs is sorely needed. The need for just such a system at East Texas State University was recognized after several incidents (such as the one described above) occurred. In order to make our resources more accessible to users and to staff, we designed and implemented a computer directory and guide to CAI programs at East Texas State.

This paper describes a method for retrieving information about CAI courses that is now being used on our campus. The Directory operates both as an on-line (interactive) information retrieval system and as a batch system. The on-line catalog allows the user to obtain information concerning a specific CAI program, a list of CAI programs for a specific subject area, a list of programs written in a specific language, and a list of CAI programs which are active or inactive. The batch catalog allows the user to obtain a list of all CAI programs or a list of those CAI programs pertinent to a specific subject area. Such a catalog system has allowed our users to become familiar with all the CAI courses residing on our system and to become literate concerning the use of specific courses.

BACKGROUND

One might assume that all CAI materials at any particular institution would be cataloged or indexed as they are developed. This, unfortunately, is not the case for three major reasons. First, initial efforts on a campus are usually spent in the area of development, rather than on documentation or cataloging. This is understandable since many initial grants and monies require that a certain level of production be achieved. Second, many of the initial courses are developed by faculty members who stay at the parent institution for only a brief period of time. Once these faculty members leave, documentation becomes dated, and these courses go unused and unattended. Third, many of the courses are developed by individuals who are novices to CAI or who are not sponsored by a specific department. Again, the documentation for these types of courses is sparse (or nonexistent) and knowledge of them is limited to a handful of computer fanatics on a campus. Problems in locating and using obscure programs make it difficult for interested users to obtain information, yet here is a case where there is significant production and a vast untouched resource.

These were problems at East Texas State University. In September 1968 a faculty research grant was awarded to begin research on computer-based education. This effort, subsequently termed the CAI project, received several grants over the next two years. By the end of the first year, a specific CAI system configuration had been defined and its major components consisting of an author language, processor, I/O routines, a controller, and student/author terminals had been selected. In-house testing then began, and concomitantly, considerable effort was expended to develop course segments. In the year that followed, experimentation with things such as terminal attributes, response time, instructional strategies, course development techniques, and student reactions to CAI occurred.

By September 1972, faculty and student demand for CAI, slow to develop in the first year, suddenly blossomed and has continued unabated. To satisfy this demand, more terminals and staff were added. A CAI Coordinator was hired whose major responsibility was to provide instructional time-sharing services for the entire University. This person maintained a contact with the faculty and with other institutions around the country and thus was in a position to offer assistance to any one desiring to implement or use a CAI course.

As interest and use of CAI continued at East Texas State University, however, no individual could possibly remember all the information concerning all the CAI courses maintained on our campus. It is significant to note, for example, that interest in CAI is broad based, effecting almost every academic discipline and practically all levels of university administration. Thus it would be foolish to assume that any individual could remember which programs do what, how they are accessed, etc.

One of the first steps taken to organize a CAI catalog was to assemble a typed, loose-leaf catalog. Although primitive by computer standards, this system at least afforded some respite from the chaos that was overwhelming the CAI users. The obvious problem with such a system was that it was difficult to update and edit information concerning specific CAI courses, especially for those that were changed from one semester to the next. Because of the limited space on our system, we were forced to rotate certain courses each semester, thus changing their status from active to in-active and vice versa. Changes of this nature were difficult to make since they required re-typing or reassembling entire pages of the catalog. This situation was obviously inadequate for the needs of our campus since it did not allow for easily modifying, expanding, or editing information. In addition, since many of the program names are not suggestive of their true purpose, it is difficult for the user to determine the status of programs without a great deal of preliminary research.

Like others before us, we discovered the inadequacies of the printed book as an information storage and retrieval device. The computerized catalog developed eliminates the need for such preliminary research, thus saving users valuable time and effort. Users are now able to locate information for a specific subject area and for a specific course; they can determine the status of a course by simply entering the corresponding function. The need for this tool is especially acute since the volume of CAI courses developed and processed during any one semester makes it imperative that our users be constantly informed of program status.

DESCRIPTION OF THE CATALOG

The CAI catalog developed at East Texas State University was specifically written to accommodate both administrative and user needs. The catalog program, called CA01, is a data base management system capable of processing both numeric and text information. It was designed so that a user with little or no training in programming can enter and retrieve information through convenient input-output modes. The CAI catalog was designed around three major objectives:

1) Data entry and report generation are controlled by the user and can be tailored to individual needs.
2) Processing is performed both interactively and by batch providing both speed and flexibility.
3) The system is powerful, yet simple to use. No extensive training is needed, only some elementary programming skill and the ability to type.

The CAI catalog is organized around four separate files; one contains the course descriptions, one stores the subject listings, another which specifies length, type and status of courses, and a fourth stores the cross references for the subject listings. For search purposes, a keyword (or function) is entered at a terminal. The files are then searched according to the keyword or combination of keywords, and the information is then generated. The various functions available to users can be used to add, delete, and update a course, enter a course under a specific subject category, retrieve information according to course name, subject heading, status, length, or type. Search commands for the catalog as well as report contents can be specified by the user in relatively simple, user-oriented language.

BUILDING THE DIRECTORY

Data entry into the directory can best be explained by the following illustration. Suppose that one has a course that needs to be placed in the catalog. The user would enter the proper mode and type:

```
    Add/(name of course)
    EXAMPLE:
       ADD/scale
```

The following category names would then be displayed on the screen:

```
    AUTHOR, TYPE, STATUS, LENGTH,
       SUPPLEMENT, DESCRIPTION
```

The user would then proceed in a fill-in-the-blank fashion and type the necessary information into the computer. The program provides a maximum of 18 lines for the course description and other necessary documentation. After pressing the ENTER key, the above information is then stored in the proper files. The course is then placed under the proper subject heading by typing:

```
    ADS/(program name)
    EXAMPLE:
       ADS/scale
```

The CAI staff assigned subject headings to the courses, using the list that appears in A. Wang's <u>Index for Computer Based Education</u> as our reference. There are no limitations on the number of subject descriptors that can be assigned to a single program. The user may specify as many subjects for a single program as he desires. For this reason, programs of an interdisciplinary interest are not confined to indexing in limited numbers of categories.

Once the program has been entered into the course files, corrections can be made easily. The user simply types:

```
UPDATE/(program name)
EXAMPLE:
   UPDATE/bio
```

The entire entry including author, type, subject, etc., is then redisplayed on the screen. At this point, the user simply spaces over to where the error or deletion occurs and makes the necessary changes. It is important to note that a user does not have to re-type the entire line or description if he wishes to replace a single character. This feature has greatly eased the burden of making corrections to courses that occur on a repeated basis, such as changes to status of programs.

To retrieve the data base, the user ordinarily signs on to a terminal, calls CA01, specifies the desired function, and immediately receives the results at the terminal. Functions include specifications for a single CAI program (i.e., course description, type, and status), a list of CAI courses for a specific language, or a list of all CAI programs on-line or off-line. For example, to request information about a course called psy, the user would type:

```
SEE/psy
```

This command would cause the following information to be printed on the user's terminal:

```
 see/psy
CAI COURSE DISPLAY --  PSY
 AUTHOR:  Eileen Rees (E.T.S.U.)
 TYPE: CW3   STAT:  ONLINE   LENGTH:
300 MINUTES   SUPPLEMENT:  HANDOUT
DESCRIPTION:
CAI lesson to be used by students en-
rolled in psychology courses for the
purpose of promoting a better under-
standing of basic concepts in psychology.
Units available are:  Piaget's theories
on mental development, listening to
students more sensitively, reinforcement
in the classroom, vocational guidance of
adolescents, psychology and the scienti-
fic method, experimental method, basic
concepts of learning, behavior modifi-
cation, schedules of reinforcement, bio-
logical bases of behavior, intrinsic and
extrinsic motivation, concept of rein-
forcement.
FUNCTION:   SEE/PSY
```

FIGURE 1

EXAMPLE OF CATALOG REQUEST FOR A SINGLE COURSE

A user may also request a list of CAI programs for a specific subject area with the command sub/(name of course). The following example illustrates the use of this command:

```
SUB/biology
```

```
 sub/biology
CAI COURSE DISPLAY --
  AUTHOR:
  TYPE:          STAT:
DESCRIPTION:
BIOMED                 MOLGEN
BUFLO                  NEURO
CLIMAT                 PH
CLOUDS                 PHOSYN
COEXIS                 POLUT
DROS                   POP
EVOLUT                 RATS
FLIES                  STERL
GANGK                  TAG
GENET                  USPOP
GENE1                  WATER1
GRAZE                  WATER2
HARDY
LIMITS
LINK
LOCKEY
MALAR
MEMBR
FUNCTION:   SUB/BIOLOGY
```

FIGURE 2

EXAMPLE OF CATALOG REQUEST FOR A SPECIFIC SUBJECT AREA

The following codes request a list of all programs written in either BASIC or COURSEWRITER and a list of all the CAI programs on- or off-line:

```
TYP/CWIII or BASIC)
ONLINE
OFFLINE
```

In addition to the on-line catalog, users may request a hardcopy listing of the information contained in the directory. Users may submit a written request to the Computer Center specifying that either a list of all CAI programs, or a list of CAI programs for a specific subject area be printed. Output, of course, is alphabatized and arranged under subject headings.

CONCLUSION

The basic elements of the CAI catalog as described above have been completed and have been actively used for the past six months. Preliminary reactions from faculty and student users of the finished components of the catalog are most enthusiastic. Although the initial creation of all the files required a certain amount of drudgery, once completed, computing services at East Texas have been made simpler, easier, and much more enjoyable.

There have been several advantages that the CAI catalog has brought to East Texas. First, information in the catalog can be changed at any time. Prior to the catalog's existence, a change in the status of a program during a semester (i.e., a program might be taken off the system) might go unnoticed. This caused problems for a faculty member who might assign a program to his class, only to discover that the program had been deleted from the system.

A second advantage of the catalog is that aggregate reports can be obtained at any time. Although a printed, updated version of the catalog is distributed to every academic department each semester, individuals may request a copy of the CAI catalog at any time. Our system is also capable of handling a request for course descriptions for a specific subject area.

Finally, the information contained in the catalog can be retrieved by the most unsophisticated user. Access to the system requires that the user know nothing more than the English language. It has been the policy of the CAI Lab that the greater the ease with which information about the system can be obtained, the more the system will be used. And, the greater the degree to which the information is accessible to the average student and faculty member, the more effective the facility will become in meeting user needs. No longer do individuals have to be a part of a select group in order to get information concerning which CAI programs are available and how they may be accessed.

Ready access has been especially important to faculty in their roles as academic advisors. Many of our students need career counseling before they can declare a major. In order for these students to become familiar with some of the various aspects of the profession, an academic advisor will often recommend that he or she take one of the CAI courses in a particular field. In this manner, the student can find out for himself whether he has an aptitude for a particular area of studies. Therefore, it is vital that the academic advisor be able to determine which courses are available and how they may be accessed.

The needs of East Texas State University are similar to the needs of other institutions. We've made an information retrieval system similar to library systems but instead of describing traditional documents (articles, books, etc.) our data base contains citations to and descriptions of CAI programs. Such a catalog ought to include all software, not just CAI materials, and we are working on just such an expansion. Accurate information must be provided in order to maintain an institution's effective posture and to insure that the needs of our users are being met. We believe that we have partially met the needs of our users by providing them with a CAI catalog system that allows them to obtain information concerning programs. We further believe that other institutions, using this system as a model, can implement similar CAI catalog systems, and that they too can provide their users with a much needed service.

A SUCCESSFUL COMPUTERIZED MANAGEMENT SYSTEM FOR INDIVIDUALIZED INSTRUCTION

Gary W. Stoaks
Computer Occupation Department
WARREN OCCUPATIONAL-TECHNICAL CENTER
Golden, Colorado 80401

ABSTRACT

One of the major problems associated with individualized instruction is the handling of the vast amounts of paperwork and recording of grades for all the various learning activity packets or modules needed for a specific course. At the Warren Occupational-Technical Center (WOTC) in Golden, Colorado, a special computerized management system has been developed that allows the full potential of the individualized instruction system to be used without tying up the course instructors in time-consuming, record-keeping tasks. This paper discusses the development of the Stoaks Autograd System and how it has been successfully implemented at the WOTC.

INTRODUCTION

In 1973 the Warren Occupational-Technical Center opened as an extension of the 12 academic high schools in the Jefferson County School District. Under the direction of the Colorado Individualized Instruction Consortium Project (CIICP), an individualized instructional program was developed at WOTC, using a Learning Activity Packet (LAP) module system.

Under the LAP system, students can proceed through a two-year program as quickly or as slowly as they need. There are, of course, checks and balances to be sure a student doesn't misuse the system.

With this self-paced system it is possible to have a class of 18 students take 18 different tests during one class period. Since a course can have between 200 and 300 LAPS, the instructors need room in their grade book for each LAP title, grade, percentage, number of points for each LAP (points are accumulated for high school class credits), date and quarter completed.

Since the successful conclusion of the CIICP in 1975, the WOTC has continued its innovative development with new improvements and progress to the total system.

Both instructors and students showed enthusiasm about the individual instruction approach, but unless a more proficient management system was developed, the chances of a totally successful system at the WOTC was questionable.

THE STOAKS AUTOGRAD MANAGEMENT SYSTEM

The department head of the computer occupations classes at the WOTC felt that it was inconsistent to teach a class in computer technology, then not apply to the classroom what was being taught. He believed this program area had the necessary resources to develop a management system that would correct the problem.

The auto body instructor and the small engine instructor became so interested that they convinced the computer instructor that their programs should be included in the initial field tests.

Nine weeks after work began, with the help of some of the computer class students, a few instructors, and the enthusiastic support of the administration, the design of the Stoaks Autograd Management System was completed.

Autograd's prime function was to release the instructors from the time-consuming process of keeping track of each of the LAP module. With the Autograd a computer card is available for every LAP the students take throughout the course. These administrative detail computer cards are prepared shortly after the students sign up for a particular course. They include the student's ID number, name of the student, the LAP code number, and the LAP title.

The cards are then placed in a special file available both to the instructor and the students. When a student completes a LAP, the student pulls the detail card for that particular LAP from the file and gives the card and the LAP test to the instructor. If the student passes the LAP test, the instructor then records the grade and signs the card.

Once the detail card is completed and signed by the instructor, it is sent to the data entry class for key punching. In the computer classes the students must punch their own cards as part of their class duties.

Once a week, or as needed, all the cards are run through the computer. Computer printouts are run with or without the students' name. These printouts are then available for the students to check. By using their ID numbers the students can use these computer printouts to give them an accounting of their progress on a week-to-week basis. Also, these computer printouts can be taken home to their parents. This sheet also lists the number of days the student was absent, the total points accumulated, and the amount of school credit earned.

Only successful course completion is recorded with this system. Credit points are rewarded for each instructional objective successfully completed and the stigma of failure does not occur as no credit is given until the student successfully completes the LAP. Students soon develop a positive and enthusiastic attitude toward their instructional content. This reduces or eliminates

© 1979 by Gary W. Stoaks

disciplinary and attendance problems and allows the instructor more time to spend with the individual student.

The self-paced aspect of the LAP requires a modification of the way school credits are earned. Under the old system, the student would be given 3/4 of a credit for each quarter the student attended a particular class. This does not work under the LAP self-paced system, because some students finish the full six-quarter course in just four quarters and other students need nine quarters or more to finish the same course. The Autograd system gives points for each LAP completed. The following chart shows how the points and credits are figured:

6,000 points earned = 6 credits earned
3,000 points earned = 3 credits earned
1,000 points earned = 1 credit earned
750 points earned = 3/4 credit earned
500 points earned = 1/2 credit earned
250 points earned = 1/4 credit earned
249 or less points= no credit earned

The balance of the student's points earned but not used are carried to the next grading period. This system allows a student to get full school credit for a course even if the student completes it before the average 8 quarters. More importantly, it allows partial or fractional credit for a student who needs more time. Some students are ready to go onto the job market much sooner than June 1st when thousands of graduates are looking for work.

Under the LAP system, no student should fail. If the student doesn't pass the LAP test, the instructor finds out why the student had problems. Solutions include counseling, special help for the student, or even re-writing the LAP. The LAP is then repeated or the instructor works with the student until the student understands the material and can pass the objectives of the LAP.

Unfortunately, there is a small percentage of students who will not pass no matter how effective the LAP system or how sincere the instructor. Here again, the Autograd system was utilized to develop an equitable drop process.

When a student is accepted into a particular program, both the student and the parents sign a statement that indicates that if, at any time, the student falls 750 points behind the class average, the student is automatically dismissed. There are two major check points at 250 and 500 points. The average is a calculation of the averages of 90% of the students that have completed the course from the time the course switched over to the LAP system.

Since the Autograd system quickly indicates when a student is starting to have problems, the instructor can immediately start working with the student on an individual basis to correct any deficiencies. Every effort is made to ensure the success of every student, but the drop process puts a share of the responsibility on the student. Of course, exceptions include illness, handicaps, and other valid reasons.

FIELD TESTING AND IMPLEMENTATION

The Stoaks Autograd System was field tested for one quarter in the computer, auto body, and small engine classes. Minor problems were quickly corrected and Autograd is now used in 19 of the 27 programs at the WOTC. The remaining 8 programs either have not completed their LAP program or prefer to remain with a traditional lecture format.

One hoped-for improvement is the replacement of the computer cards with a key-to-disk remote terminal input. With this system, a teacher can put new information into a student's record immediately. Not only will this new system benefit the teacher, the students can use their student ID number to review their total class progress at any time.

SUMMARY

A successful individualized instruction approach requires more than just a successful LAP system. The development of a proficient management system is equally important. The Stoaks Autograd System is such a system.

COMPUTER ENHANCEMENT OF CULTURAL TRANSITION

Edward J. Gauss
Computer Science
University of Alaska
Fairbanks, Alaska 99701
(907) 479-7332

Introduction The indigenous people of the Alaska bush, the Indian, the Eskimo, and the Aleut, are undergoing a complex cultural transition. The steel-edged tool, the airplane, the rifle, and the outboard motor are among those items which have created a cultural dependency and produced a distrubing posture of inferiority. Our educational system contains many concepts not found in the indigenous culture and our efforts to educate often intensify the dumb native feeling. There was no need for reading or for arithmetic in a hunting and fishing society.

After nearly ten years of work we feel confident that the use of computing machines is of profound positive benefit, not only to stimulate learning in the bush but also to facilitate positive cultural transition. Three important results are reported here: (1) How FORTRAN in Nome removed the self-imposed cultural isolation from an Eskimo child. (2) How interactive computing in the Indian village of Nulato circumvented a cultural barrier which hindered teaching. (3) How handhelds in the Eskimo town of Dillingham were used to achieve arithmetic literacy.

I. THE BUSH

Geography Alaska has both the largest land area and the smallest population of any state. The majority of the population is concentrated in the railbelt and the southeastern panhandle. The remaining region, the bush is sparsely populated. Its population is contained in several hundred tiny villages of which only a very few have more than one thousand people. A population of three hundred is a large village. Timely transportation is by single pilot aircraft. Telephone service is limited or does not exist. The village school teacher may be the only winter resident not of native blood. The school may provide the only payroll of the community.

Village life is missing certain factors that facilitate education. Homes are tiny cabins dominated by eating and sleeping functions. There is no place in the home for reading nor is there any tradition of reading in the home. There is no study desk where arithmetic problems may be worked. The home activities provide no context for the use of reading or arithmetic. These are things done only at school.

Bush Mathematics There was little need for computing in the early culture. Distribution was by share and take. A moral injunction against greed and right of kin to take according to their need eliminate any need for accounting. The storekeeper changed this system. He must sell, not give, if he wishes to stay in business. He must then enforce arithmetic in the community. A housewife may select her goods and take them to the counter. She then lays down all of her money. The storekeeper counts the money and totals the goods. He may return change, or he may select goods to be returned to the store, depending upon his totals. The housewife does not understand the process. We can only hope that the storekeeper maintains a consistent standard of arithmetic. It is unclear, especially in the mind of the native, that this is always true. There is a feeling that all storekeepers cheat and use something called arithmetic to keep you from taking what you want. There is no belief that two people can add up the same list of numbers and obtain the same result. Arithmetic does not thrive in such a culture.

Schools Only eighth grade education has been offered in most of the villages. Many of these have only one room school houses, and there has been a problem with teacher inexperience and retention. Post primary education was confined to a few communities with populations of more than a thousand. Boarding facilities were provided and have not been fully satisfactory. There is a very strong family tie and village identity. At boarding schools student depression has been a problem and suicide has resulted. An effort is now being made to introduce secondary education into the smaller schools but no one knows how successful this will be. What can you expect from a one room high school?

II. RESOURCE

Start Up This program was started while I was Director of the University of Alaska Computer Center. Reporting to the Research Vice President, I had sufficient freedom to initiate such projects. Several formal proposals for computer work in the bush had been prepared by various agencies, but the sparseness of the population, the high transportation and equipment costs made their cost effectiveness prohibitive. As Director I could lend a keypunch to Nome, find air freight to get it there, provide computer time, and use the project as a training activity for my student assistants. We operated an IBM 360/40 and a Data General Nova. Viavant, of the University of Utah, has observed that my owning a bush aircraft and using it to transport computing to the bush contributed materially to the success of the project.

High Schools It was essential to locate high schools where a teacher was interested in using computing in the classroom. Four locations were found and they formed an excellent gradation from the city to the bush. The first was in Fairbanks where the University is located. It had a 1970 population of 14,771 and was the second largest city in Alaska. It belongs to the dominant cultural system, and the school population is less than 7% native. In Fairbanks we learned that the University could assist in the teaching of FORTRAN to high school and junior high students.

Nome was our second location and its 2488 population is predominantly Eskimo. It has an active tourist trade and the residue of a fifty-year old mining industry. It was first a whiteman mining town before the the Eskimo population was attracted to it. Acculturation is strong although the students at the boarding high school have many problems with cultural transition.

Dillingham is also Eskimo with a population of 914. It is a fishing town with much less obvious white influence than Nome. Its high school is attended by many students who come from the smaller villages.

Nulato only recently obtained a high school. It population of 304 is Indian. There are no boarding students and the school is the largest, if not the only, employer in the village. The kinfolk system interferes with transactions at the general store reducing the incentive for arithmetic literacy. The only people who are not native are associated with either the church or the school.

III. WHAT WAS DONE

FORTRAN in Nome In 1971 Dave Eskeldson, a Nome teacher, returned from San Jose with an interest in using computing to motivate mathematics learning. There is little in the Eskimo world that demands or relates to mathematics. I was very enthusiastic and promised him support. A keypunch was shipped to Nome and installed in the classroom. There, students could learn how to program in FORTRAN as a bonus activity. Experiments with the mail system told us that the best we could hope for was one week turn around between Nome and the University. That delay could be a very frustrating thing for debugging. Yet debugging is an essential part of learning to program. Eskeldson required his students to proof read their cards before they were submitted to him. He then proofed them a second time. They were then bound with control cards and sent to the University. My student assistants acted as secret gnomes and expedited the processing and return of the Nome batches. I had them proofread the output and correct and resubmit any control card or card out of order problems. Total control of program content was left to Eskeldson.

There was one Eskimo boy in Nome who makes an excellent study of what the program achieved. There seemed to be a dotted line around him in the classroom. Inside was Eskimo territory and values, and that did not include mathematics. Outside was the classroom. As long as there was no trespass, he was quite content to exist in the classroom and stay as much as possible within his territory. One of the early lessons was to write a FORTRAN program to have the computer print out the student's name. The boy understood that the computer would print out any word of his selection. So, he selected a nice four-letter Anglo-Saxon word. He sent his cards to us, and my gnomes for Nome were a bit uneasy and asked if they should edit. I felt that this was Eskeldson's decision, not mine, and the word was more offensive in the University student culture than in Nome street culture. The printout went back, and in the middle of the green and white striped paper was the four-letter word. That was fun! It was suggested that the student learn a little more. What could be done with a DO loop? My gnomes were uneasy to see page after page of the offensive words come off the computer. Arrays were now introduced and different words of the same cultural value now appeared in pretty patterns. About this time the dotted line of territory disappeared. The four-letter words were no longer the objective, the learning was the objective. The kid was hooked! It is reported that he became overprotective of the keypunch, the only physical evidence of the computer in Nome. He insisted that other students clean up. He was now a participant in the educational system instead of an involuntary visitor.

Remote batch work was extended to include Perry Atkinson's classroom in Dillingham and

Royce Purinton's classroom in Nulato. Peter Woodring, an electrical engineering student on my staff had to develop a power protection device to protect the keypunch we sent to Nulato. Low voltage there caused motors to burn out. It is interesting to note that these keypunches ran for nearly five years without maintenance. The bush teachers lubricated the machines and did repairs not normally made by the user. The program continued until reorganization at the University left it without a sponsor.

The most important result of the remote batch work was not learning FORTRAN. There was only so much you could do with a turnaround that sometimes was two weeks. The important thing was the discovery by the students that they could subdue the computer, that incredible symbol of the dominant culture, and compel it to do their bidding. This is very important in bush education as it is very easy to stimulate the response, "I'm just a dumb native" which totally blocks learning. If the native learned to cope with the computer, he could have more confidence that he could cope with the dominant culture in other areas.

Charlie's Dog Many of my computer center gnomes became quite interested in bush education. We had a time-sharing Nova available so we decided to take a terminal to the bush. Orvick, of our Rural Education project, had suggested that reading retention was the biggest problem of bush education. The child will structurally read the text, but retain nothing of the content. So, we wrote of "Charlie's Dog." This was a group effort to construct primitive demonstration of computer-assisted reading comprehension. My students then wrote two BASIC programs, Teacher and Student. Teacher enabled the bush teacher to write a paragraph of material relating to village life and then asked questions about it. Student presented this material to the student and provided positive and negative prompts of increasing vigor. A story about Charlie's dog and the big dog race was written by Diane Roy, who was doing student teaching at the time. It asked culturally relevant questions such as, "What kind of mittens did Charlie have?"

Communications to the bush were not good. We planned to use dial up telephone lines. At that time there was a 25 db loss between the University and the toll connection. Microwave carried the channel to the Anchorage toll center and then it went by satellite relay to Nome. The echo suppression was poor and often the lines were unsuitable for conversation of other than the most labored type. I first sent just a modem to Nome and Dillingham. Peter Woodring bounced signals off of them and felt that it was worth a try.

A Texas Instruments 700 terminal was loaded in my ski-equipped airplane and I flew to Dillingham. In Dillingham we patched into the school phones and tested out our equipment in the evening. Everything worked well. There were only three phone lines available to the school; in cooperation with school officials, we selected one that would least distrupt the school regime. It turned out that we were very disruptive but in a different context.

There are only four long distance lines out of Dillingham and we planned to tie one up for most of the day. We took a line at 8am and planned to hold it as long as we had students. We had had a little trouble getting a good circuit but got the toll operators' (located in Anchorage) cooperation after a careful briefing and explanation of our project. About 10am the line dropped for no apparent reason. We re-established the circuit and lost it again in ten minutes. Leo Lutchansky of the Dillingham schools determined that the local telephone repairman was bothered by one of his long line circuits that seemed to be hung up. He monitored it for a while and then decided that it must be some sort of test tone coming from the RCA board. After a while, he decided that they had no right to tie up one line like that. Exercising bush justice, he cancelled the circuit. When we came back on, he cancelled us a second time. He was very angry at RCA by the time Leo called. Leo explained everything and we had no further problems. (It should be mentioned that at the RCA board in Fairbanks a former student of mine, Perry Stoop, was monitoring our experiment and helping to keep our circuits intact.)

We demonstrated Charlie's Dog, played Star Trek, and wrote a few programs in BASIC for groups from math teacher Atkinson's class.

The older students were fascinated by Charlie's Dog and wanted some more stories. I invited them to write some, and they did. These student-written stories were to prove the most exciting when we went to Nome. Around 5pm we had to shut the terminal down. Students had stayed after school just to play with the computer. Atkinson had observed that the game of Star Trek had taught angle addition and subtraction to his students who had resisted his earlier teaching efforts.

Nome school combines grammar school and junior high. I was concerned that Charlie's Dog, because it required use of the keyboard, might falter with the younger Eskimo children. The phone had a bad echo on it and I spent that evening line testing and doing some limited demonstrations for teachers.

Like Dillingham, the Nome students were fascinated with Charlie's Dog. It was presented to an elementary school class. They clustered around the terminal and they helped each other find the needed keys to spell "caribou." They liked even better the stories written at Dillingham and then wrote a few of their own. (One was a fantasy about a Nome teenybopper who went swimming in her fur bikini after cutting a hole in the ice.)
Because of weather I spent an extra day in Nome and it was filled with students seeking to

use the computer. I received several inquiries from teachers asking if I could leave the equipment because it had generated the highest interest of any teaching devices of their experience. I answered that while I could leave the equipment, the phone charges had probably exceeded the budget of my HEW grant and would probably cost them $200/day. This would have been beyond the means of the school.

The Last Voyage of Charlie's Dog Dr. William Viavant of the University of Utah accompanied me on our last bush tour. The IBM Corporation had loaned us a 5100 computer and Viavant had translated the BASIC programs *Teacher* and *Student* into APL. This was the first time any computer had been brought to Nulato. There we landed the plane on the river near the school. The computer was loaded on a dog sled pulled by a snow machine. We took it to the school and allowed it to warm up. We had talked carefully with IBM about the environment for the 5100 and they were optimistic about everything but the effect of cold upon the tapes. Nulato was known to have power which fluctuated widely in voltage (from 80 to 125 volts) and frequency of 55 to 65 cycles. The computer was allowed to temperature stabilize and it worked well. We had the expected trouble with the tapes.

In Nulato we discovered another benefit of using the computer in bush teaching. I had visited Nulato several times and had taught in the classroom. I was bothered by the total lack of visual response from the students. Royce Purinton, the math teacher, had explained to me that it was impolite in the local culture to show reaction, especially to a stranger. There was no way I could tune my presentation to optimize audience reaction.

While Viavant helped some students play with APL, I noticed that when the students looked at their work on the computer screen, personality came back into their faces; I could read the degree of interest or misunderstanding from their faces. The students did not have the tabu against expression to the computer! Yet when any of the teachers addressed them directly we received no facial response. The computer had made a cultural bridge that none of us had been able to make.

Viavant was, in essence, lecturing through the computer, and the students were responding to the computer not knowing that it was really Viavant. Yet Viavant could steer his presentation on the basis of the response they expressed to the computer.

Handhelds and Arithmetic Perry Atkinson in Dillingham faced the problem of arithmetic literacy in his students. Boarding students from the bush schools often arrived at the high schools with third grade skills in arithmetic. They could only add and subtract pairs of single digit numbers. The idea of a unique total for a grocery bill was foreign to them. If they added the list they would get one figure, their neighbor another, and the teacher still another. The teacher's figure had to be accepted, not because of any sort of mathematical law, but because the teacher had greater authority. These students had been drilled in arithmetic by every arithmetic teacher since third grade and they still had made no progress. Perry decided then an eyeglasses approach was needed. He declared that he was going to teach the use of arithmetic and let hand-held calculators perform the mechanics of arithmetic.

He obtained a dozen handhelds. If the student passed the course, as a reward he was to be given the calculator. One does not provide eyeglasses and then take them away at the end of the semester. He then taught arithmetic using handhelds. The students made the profound discovery that they could add up a grocery bill and obtain the same total as their neighbor and that was identical to the total provided by the teacher! By the end of one semester, the students could use arithmetic for the needs of a mail order purchase and the like. By realizing that arithmetic was an exact science, and not a question of greater authority, they began to accept many other tenants of our commercial culture.

Retrospect In the bush you often go forth and leave new tracks in the snow. The next day the wind may blow and cover the trail with snow. But sometimes you look back and see others following the trail through drifts and improving as they go.

The use of computing has been firmly established in the Fairbanks schools. My university students complain that they must compete with the high school students who come to the University in the evening to use the terminals. While the high schools are purchasing computer time they are restricted in the number of on-site terminals. The trail is less established in the bush. Both Dillingham and Nome continued the computing programs for a while after University reorganization eliminated sponsorship. They had developed funds to continue, but without the fringe encouragement from the Computer Center (such as the computer gnomes) the trail became blown with snow and little used. Furthermore, Nome School was reorganized, placing mathematics in a mix with team teaching in an open classroom. FORTRAN did not thrive in that mix. Recently Eskeldson has been transferred to the high school, and he and I are looking at home computers for purchase by Nome to reopen the trail.

Atkinson's handheld program in Dillingham has drawn hostility from traditional mathematics teachers. They say he is teaching the use of a crutch. With this epitaph, handhelds are declared bad. The handheld program requires two items for its continuance. First, it must be accepted by the school board to insure continued funds for the calculators. Second, the dependency of its graduates upon the handhelds must be accepted by other teachers in their classrooms.

Nulato, through the activity of Purinton, purchased a Wang computer, and it has been used there both in the high school and in the extension programs of the University. While Nulato is not without administrative change, something positive is happening there. Most native students have not gone on to the university. Yet an unusually large number of Nulato High School graduates go on. Most of them attend the University of Alaska and show up in my freshman course, Computers and Man. Charlie's Dog has proven to be a good lead dog!

COMPUTING IN MINORITY INSTITUTIONS: 1976 - 1977

John W. Hamblen (Moderator)
University of Missouri-Rolla
Rolla, MO 65401

Clinton Jones
Fisk University
Nashville, TN 37203

Jesse Lewis
Jackson State University
Jackson, MS 39217

Sr. Patricia Marshall*
Xavier University of Louisiana
New Orleans, LA 70125

Minority institutions: Who and what are they? Where are they, and how long have they existed? Are they a dying breed, or are they flourishing? What sets them apart? Is there anything about computing in these institutions from which others can learn? Do they have special needs?

This paper, based primarily on data from the Fourth Inventory of Computers in Higher Education (FICHE)[1] compiled by John Hamblen, both raises the questions and attempts to answer them. Additional work is in progress,[2] but most of the conclusions drawn in this paper are based on data from FICHE.

Minority institutions are defined in this study as those accredited post-secondary institutions with approximately 50% or more of their students from one or more socio-economically disadvantaged ethnic background (black, Hispanic, American Indian, Asian). Altogether the 228 minority institutions thus identified in 1976-1977 enrolled a total of 743,810 students.[3] (If the newer, not-yet-accredited Indian and Spanish institutions had been counted, the total would have been closer to 285.)

Dominating the sample drawn from the FICHE survey were the predominantly black institutions (see Figure 1), which constituted 72% of the 228. These included, at the time of the study, 105 historically black schools (mostly four-year institutions, some of whose histories span a century or more) and 59 formed more recently for a total of 164. (These latter, chiefly urban based, included a large number of two-year colleges; and at least 10% of them also included fairly sizable proportions of students of Hispanic origin.)

The predominantly Hispanic institutions included 21 in continental United States, 19 in Puerto Rico, and one in the Virgin Islands (which is also predominantly black). (Both the U. S. and Puerto Rican groups are growing in number.)

American Indian institutions, which included Eskimo and Native Alaskan schools, numbered ten at the time of the study. Most were two-year colleges, and three of the schools were in Alaska. (Counting the emerging institutions would bring the total to approximately 30, seven of which are in Alaska. This is the fastest-growing group of minority institutions.)

Institutions whose largest minority group is Asian numbered 13 in Hawaii and one in American Samoa. (Today we would also add one in Guam and one in Micronesia.)

Enrollment Profile

Since the non-black institutions made up a relatively small portion of the total minority institution sample at the time of the study (see Figure 1), little attempt has been made to differentiate among the minorities with regard to interpreting the data. (It should not be inferred from this, though, that the minority institutions are monolithic in all their characteristics.)

Of greater significance are the differences in enrollment among institutions categorized by highest level of offering (Figure 2) and by enrollment ranges (Figure 3). The largest difference in enrollment occurred (as is evident from Figures 2 and 3) between private schools (enrolling 137,623 students, or 18.5%) and public (enrolling 606,187 students,or 81.5%). Another significant fact shows up in Figure 2: more than half the students in minority institutions were in schools whose highest degree was the associate; and 91.5% of these students were in the public institutions (many of which are relatively recent in origin). It appears that the newer institutions (mostly public, two-year, predominantly black or mixed colleges) enrolled many more students than most of the older schools.

FICHE Response Rate

Minority Institutions responding to the FICHE survey were close to 51%, which is roughly comparable to the 58% overall response rate. That it was a little lower may reflect the commonly overcommitted situation with regard to personnel in minority institutions, as well as less experience with the inventory forms.

Officer to Whom Installation Reported

Nationally the pattern of officers to whom installations reported in 1976-1977 was somewhat random, and the minority institutions were not significantly different in this respect, as can be seen in Figure 4. As with all institutions, the

---

* To whom to address correspondence

FIGURE I

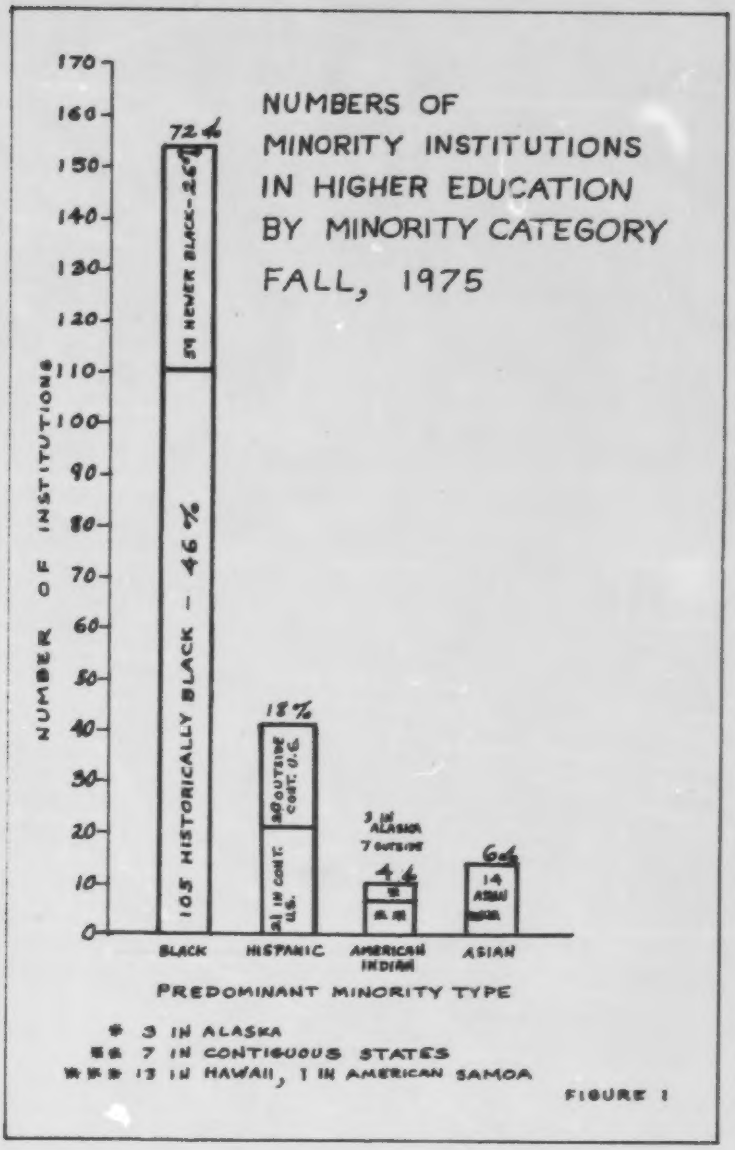
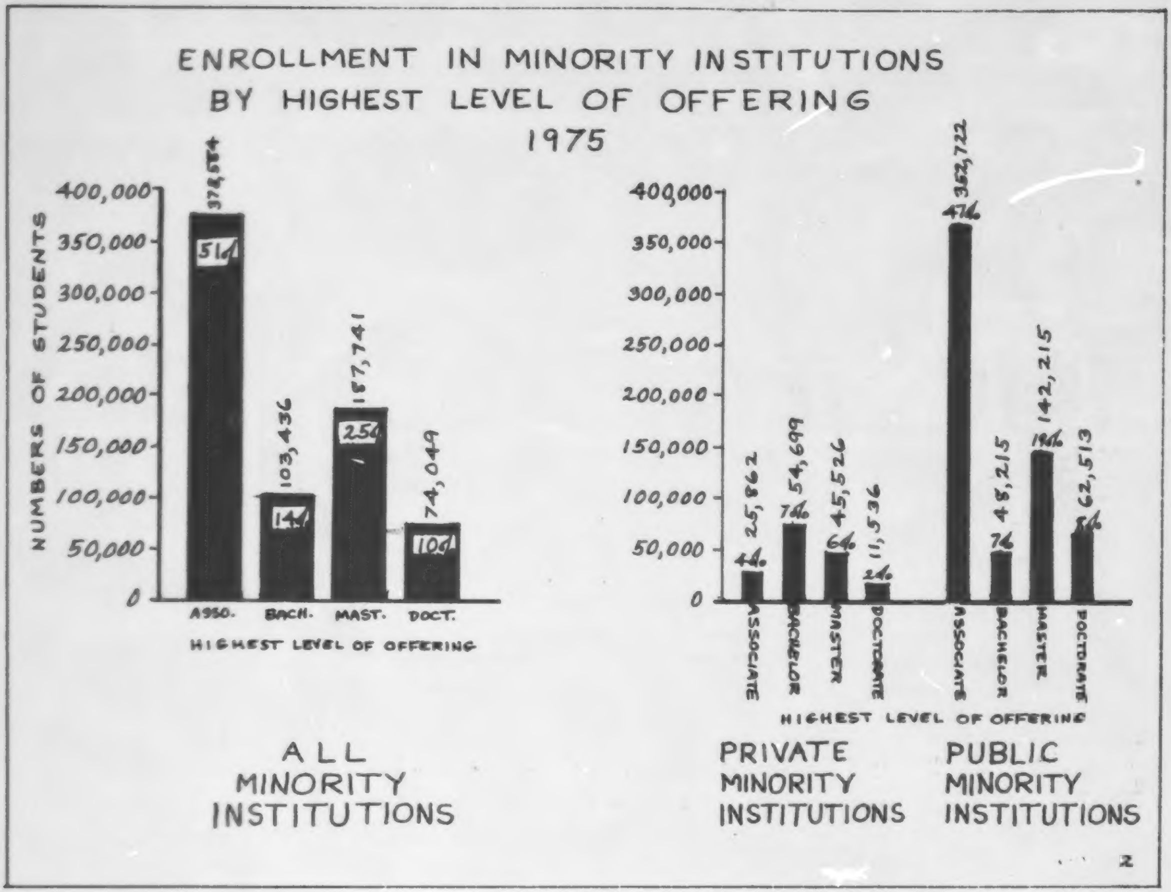
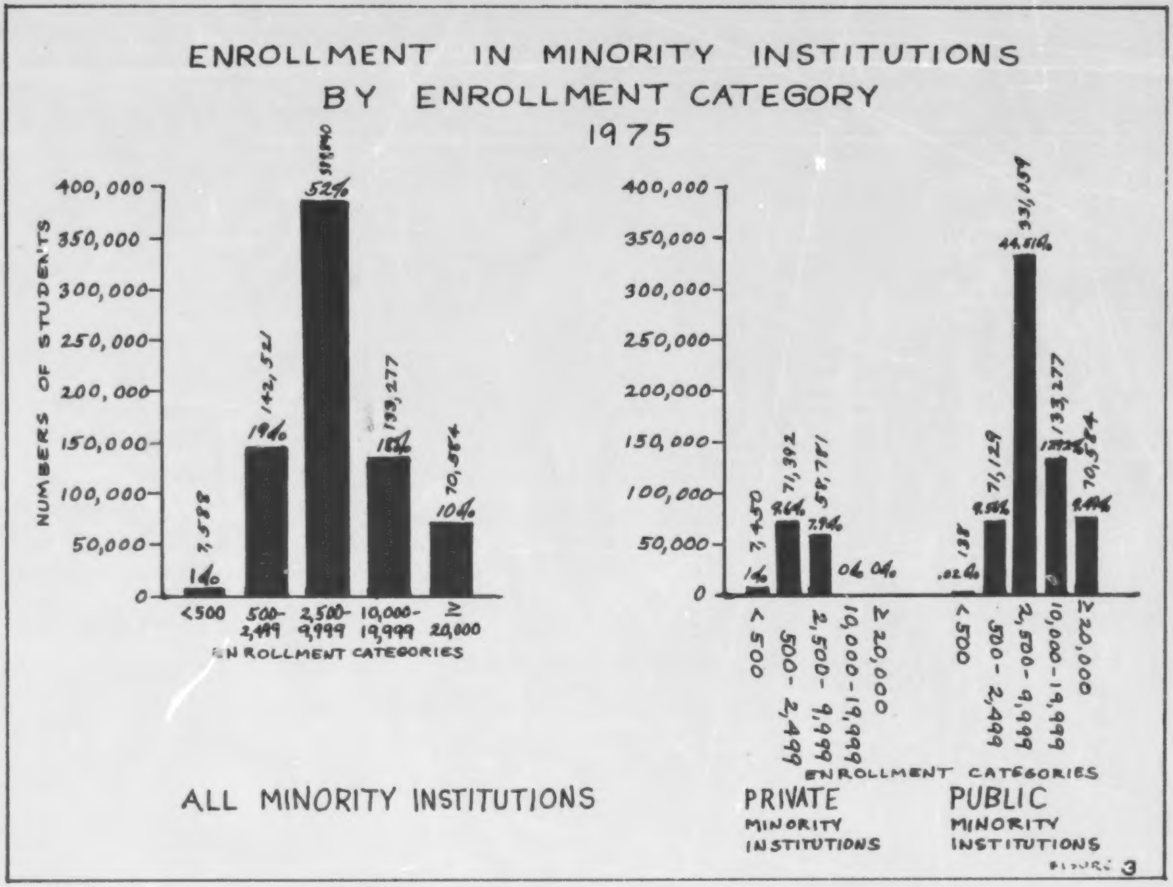

NUMBERS OF MINORITY INSTITUTIONS IN HIGHER EDUCATION BY MINORITY CATEGORY
FALL, 1975
NUMBER OF INSTITUTIONS
170
160
150
140
130
120
110
100
90
80
70
60
50
40
30
20
10
0
72%
59 NEWER BLACK-26%
105 HISTORICALLY BLACK - 46%
18%
20 OUTSIDE CONT. U.S.
21 IN CONT. U.S.
3 IN ALASKA
7 OUTSIDE
4%
*
**
6%
14 ASIAN
***
BLACK
HISPANIC
AMERICAN INDIAN
ASIAN
PREDOMINANT MINORITY TYPE
* 3 IN ALASKA
** 7 IN CONTIGUOUS STATES
*** 13 IN HAWAII, 1 IN AMERICAN SAMOA
FIGURE 1

FIGURE 2



ENROLLMENT IN MINORITY INSTITUTIONS BY HIGHEST LEVEL OF OFFERING 1975

FIGURE 3



ENROLLMENT IN MINORITY INSTITUTIONS BY ENROLLMENT CATEGORY 1975

FIGURE 4

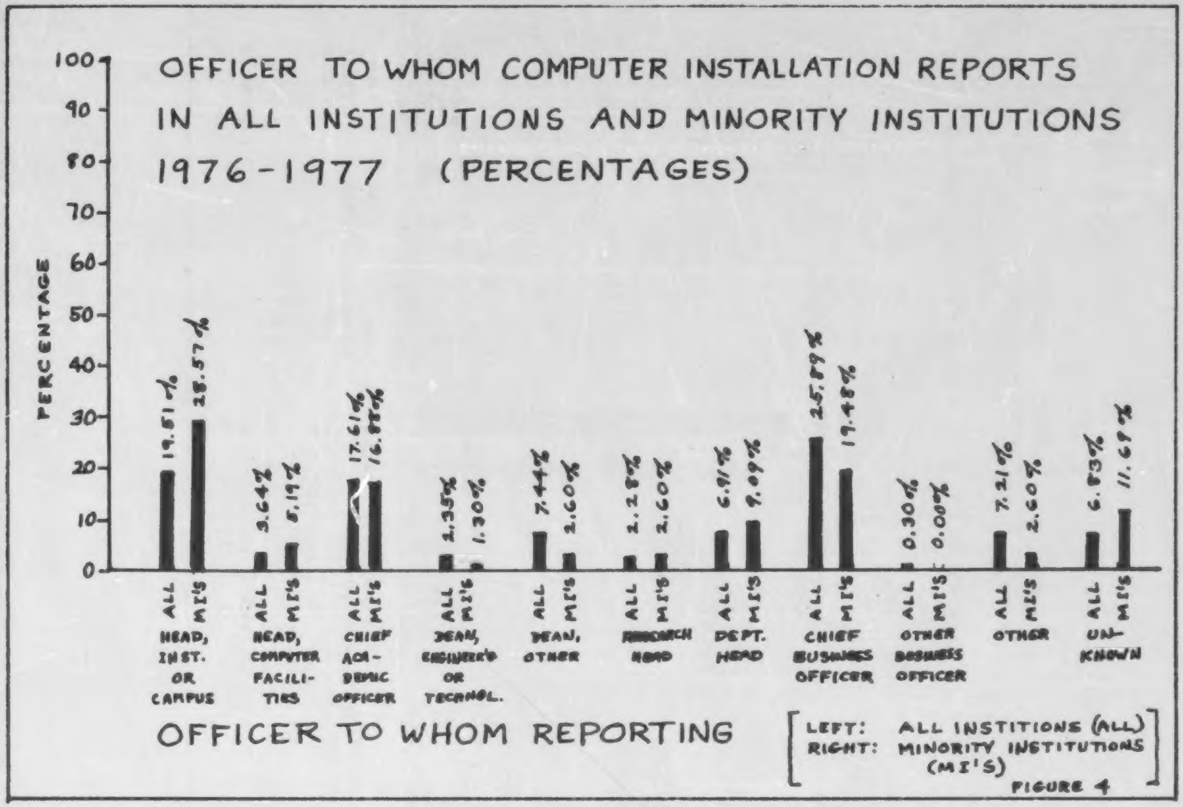
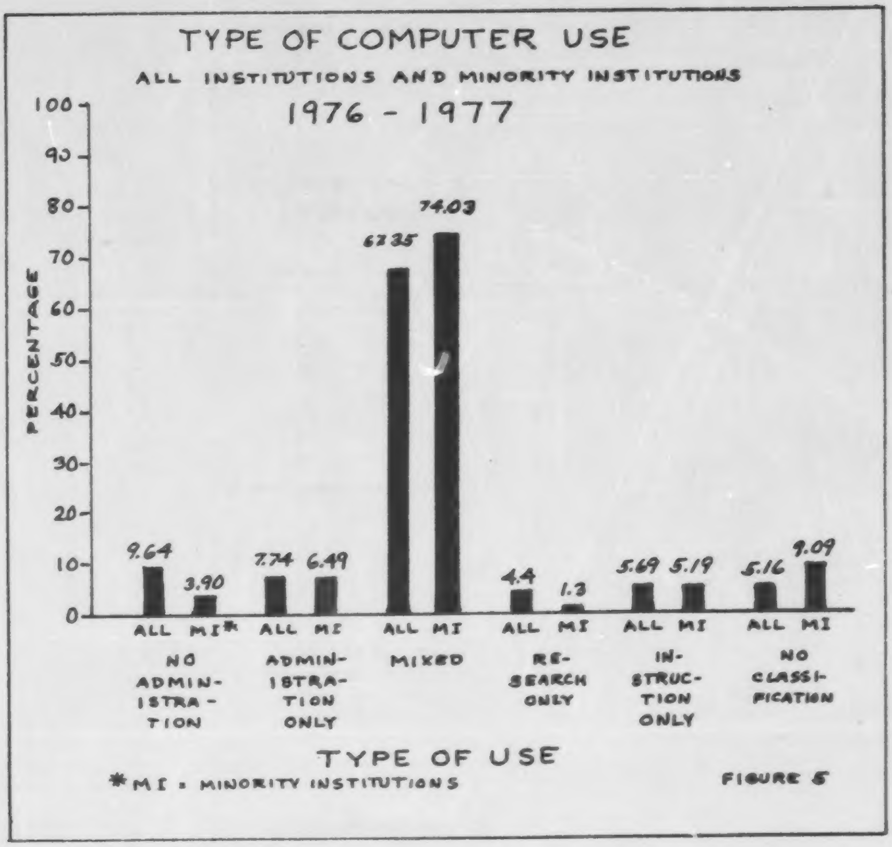

OFFICER TO WHOM COMPUTER INSTALLATION REPORTS
IN ALL INSTITUTIONS AND MINORITY INSTITUTIONS
1976-1977 (PERCENTAGES)
PERCENTAGE
100
90
80
70
60
50
40
30
20
10
0
19.51%
28.57%
3.64%
5.19%
17.61%
16.88%
2.35%
1.30%
7.44%
2.60%
2.28%
2.60%
6.91%
9.09%
25.89%
19.48%
0.30%
0.00%
7.21%
2.60%
6.83%
11.69%
ALL
MI'S
HEAD, INST. OR CAMPUS
HEAD, COMPUTER FACILITIES
CHIEF ACADEMIC OFFICER
DEAN, ENGINEER'G OR TECHNOL.
DEAN, OTHER
RESEARCH HEAD
DEPT. HEAD
CHIEF BUSINESS OFFICER
OTHER BUSINESS OFFICER
OTHER
UNKNOWN
OFFICER TO WHOM REPORTING
LEFT: ALL INSTITIONS (ALL)
RIGHT: MINORITY INSTITUTIONS (MI'S)
FIGURE 4

FIGURE 5


TYPE OF COMPUTER USE
ALL INSTITUTIONS AND MINORITY INSTITUTIONS
1976 - 1977
PERCENTAGE
100
90
80
70
60
50
40
30
20
10
0
9.64
3.90
7.74
6.49
67.35
74.03
4.4
1.3
5.69
5.19
5.16
9.09
ALL
MI*
ALL
MI
ALL
MI
ALL
MI
ALL
MI
ALL
MI
NO ADMINISTRATION
ADMINISTRATION ONLY
MIXED
RESEARCH ONLY
INSTRUCTION ONLY
NO CLASSIFICATION
TYPE OF USE
*MI = MINORITY INSTITUTIONS
FIGURE 5

three highest reporting levels were (a) the head of the institution or campus (28.5% in minority institutions, 19.51% in all), (b) the chief business officer (19.48% in minority institutions, 25.89% in all), and (c) the chief academic officer (16.88% in minority institutions, 17.61% in all). Minority installations tended to lean a little more heavily toward reporting to the head of the institution than did their non-minority counterparts. Perhaps they had had the opportunity to observe the soundness of this approach from other installations serving both academic and administrative users. (Most minority institutions using computers use them for multiple purposes.) On the other hand, the data may only reflect that large institutions in the total population can afford separate installations for their chief business officers.

Types of Use

Nearly 3/4 of the minority institutions reporting were using their computers for both administration and instruction, as well as a little research. This was not much more than the 2/3 reported of the total population (with majority institutions doing a higher proportion of research). Figure 5 shows that minority institutions followed the general pattern for the most part. It is safe to assume, though, in view of the number of two-year colleges, that not much research use occurred even in the multiple-purpose installations in minority institutions. Scoring low in type of use were research only, no administration, instruction only, and administration only. No classification was reported by nine percent. Presumably in these cases computers were still on order, or computers had been in use for too short a time for the installations to have accumulated meaningful data. These data, of course, reveal nothing of the total amount of use, its quality, the quality of access of users, cost effectiveness, etc. However, more will be said later in this paper about academic and administrative computing.

Hardware Profile

On the surface, the data on hardware on (or accessible to) the campuses of minority schools show little difference from the data on other campuses of comparable size. Nearly 30% of the minority institutions were found to have had no computing at the time of the study. For all institutions this percentage was 31%. FICHE data indicated that 40% of the minority institutions reporting supplied interactive computing and 36% also used remote computing services (to a small extent). For the non-minority installations the data showed 41% with interactive computing and 32% using remote computing services (and to a greater extent, as we will show later).

Although we are talking about somewhat small samples outside the black institutions, it is noteworthy that half the Indian schools in the sample had no computing and that of the minority institutions with no computing, all but two were private and more than 56% were black.

An analysis of hardware acquisition dates shows that nearly 2/3 of the computers on minority campuses were acquired in 1974 or later. This suggests that minority institutions acquired computer hardware using federal funding available to them in Title III programs. The MISIP (Minority Institution Science Improvement Program) program in the National Science Foundation has been limited in funds and therefore reluctant to spend money for computers, so it is safe to assume that only a very small proportion of the funding could have come from this source. A few small computers (such as stripped-down PDP-8Es from DEC) found their way into minority institutions as gifts in 1974.

The assumption that much of the hardware was acquired through Title III programs is strengthened when one looks at the type of hardware and the names of the installations. Small business sytems and installations with names such as "Data Processing Center" are rather well represented. It is reasonable to conjecture that many of these computers were acquired primarily as tools to be used in strengthening the planning, management, and evaluation capabilities of some of the developing institutions. The Title III grants require a certain percentage of funds to be allocated for this purpose.

In general it appears that minority institutions have used the federal funding recently made available to them to attempt to catch up to the majority institutions in computer hardware acquisition. Hardware is, of course, only one component in a successful system. It does not automatically follow that either administrative or academic use in minority institutions will quickly approach national levels; however, the data do suggest that minority institutions have made a beginning.

Software Profile

Table 1 compares minority and non-minority institutions in software availability. With two exceptions the percentages were nearly uniform. The two exceptions, however, are highly significant. First, no minority institutions were providing PASCAL, considered by many to be the language of the eighties and now being implemented on inexpensive microcomputers for both the hobby and educational markets. The availability of graphics was also conspicuously low on minority campuses (only 5.4%). These two differences point to lack of expertise and lack of powerful tools for gaining expertise, especially in academic computing.

The absence of PASCAL in 1976-1977 is not surprising; however, when compared with its presence on nearly ten percent of all the non-minority institutions, even then, this fact shows that no minority institutions were on the leading edge of the micro revolution, which may have contained some of the seeds of cost effectiveness so important, especially to the minority institutions. Furthermore, graphics, an educationally effective tool found in nearly 22% of the non-minority installations but only 5% of the minority installations, is another potentially large wedge in the gap between minority and majority institutions, especially in academic computing.

Computer-related Expenditures

Since approximately 3/4 of all minority institutions at the time of the study fell in the enrollment category of less than 10,000, it seems reasonable to compare data from minority institutions with data from non-minority institutions (NON-M in Table 2) in the same enrollment categories and types of institutions. The NON-M entries in Table 2 were obtained from the FICHE survey data by subtracting minority data elements from the corresponding data elements given for all institutions reporting and then obtaining the average expenditures per institution for each computer-related area based on the number of institutions responding in that given area.

It can be observed from Table 2 that minority institutions generally spent less on the average than majority institutions. This might be expected. However, significant differeneces are noticeable in certain categories. The first is that one-time capital expenditures (hardware purchases, building, and furniture) in the 2,500-9,999 enrollment range were much higher for minority institutions. Hardware purchases were almost twice as high, and building and furniture expenditures were more than three times higher. These data probably reflect start-up costs for institutions using computers for the first time. In addition, however, hardware lease expenditures of minority institutions in this enrollment category were also a little higher. This again may point to funding having come from HEW's Title III grants, which tend to encourage lease rather than purchase of hardware, except in the case of less-expensive items such as terminals and certain peripherals.

One other significant difference in average expenditures between minority and majority schools shows up in Table 2. Off-campus services expenditures were more than twice as high among non-minority institutions of 2,500-9,999 enrollment. Several factors may have been at work here. If off-campus services were commercial in origin, they would have been too expensive for institutions without an established budget for computing. Software on large machines supplying computing power in networks is often not the most cost effective, with slow response times driving up telecommunication costs. Geographic isolation in the case of some of the institutions can also escalate telecommunication costs, scaring off some would-be first-time users. Such factors as these, plus the emerging cost effectiveness of mini computers for applications in which most small colleges may be interested, or for which they feel ready, may account for the lack of popularity of off-campus services among minority institutions.

Table 3, presenting data in a different way (public and private institutions at three degree-granting levels), and including two-year institutions with enrollments up to 19,999, shows some differences between public and private minority schools. Average expenditures for hardware purchases, software purchases, and off-campus services in public minority institutions were approximately twice those of private minority institutions in 1976-1977. This table also shows that two-year public minority institutions' expenditures (in the 2,500-20,000 enrollment range) are about twice as high as for comparable majority institutions. Again this probably reflects first hardware acquisitions and start-up costs for institutions somewhat recently established.

On the other hand the private minority schools were generally well under the average expenditures for the non-minority institutions, with one notable exception (in schools offering master degrees and having enrollments between 500 and 2,499). This exception probably also reflects first-time start-up costs for these institutions. It also probably reflects enrollment more than degree-granting level, since so many of the master programs in minority schools are in the discipline of education, where computers are still not extensively used (these types of expenditures will be examined later).

Finally Table 4 compares total computer-related expenditures over a three-year period (Fiscal 1975 through 1977) and shows federal income in Fiscal 1977, with a projection of Fiscal 1978 expenditures. Two significant facts emerge from an examination of the data in Table 4. First, average computer-related expenditures in minority institutions (public or private) were considerably less in all three years than those of non-minority institutions. Second, the rate of increase in expenditures from year to year increased in the minority institutions (especially public), bringing them up to 3/5 of the level of the other institutions in Fiscal 1977 from a level less than 1/2 in Fiscal 1975. This again probably reflects start-up costs, given that one of the components of the total expenditures was hardware and other capital items associated with beginnings. If this is so, this fact does not indicate a narrowing of the gap between minority and majority institutions because hardware is only one of the ingredients necessary to do that, and it represents only a one-time cost.

Federal funding for computing in the minority institutions in Fiscal 1977 was, if we include all the costs, roughly less than 1/3 that for non-minority institutions on the average. This fact is significant when matched against the fact that much of the Title III funding for developing institutions was at that time going to minority institutions. In other

TABLE 1

PERCENT OF INSTALLATIONS IN REPORING INSTITUTIONS
(MINORITY AND NON-MINORITY)
PROVIDING VARIOUS COMPUTER LANGUAGES IN 1976-1977

LANGUAGE	% INSTALLED IN MINORITY INSTITUTIONS	% INSTALLED IN NON-MINORITY INSTITUTIONS ENROLLMENT UNDER 10,000	% INSTALLED IN ALL NON-MINORITY INSTITUTIONS	% INSTALLED IN ALL INSTITUTIONS
BASIC	60.4	52.6	54.4	54.8
APL	23.0	17.2	19.6	19.8
COBOL	69.2	58.9	58.6	59.2
PL/1	25.2	21.5	25.5	25.4
PLANIT	0.0	0.6	1.0	0.9
LOGO	0.0	0.4	1.0	0.8
COURSEWRITER	6.6	5.1	5.7	5.7
FORTRAN	73.6	68.3	71.3	71.5
GRAPHICS	5.4	15.7	21.8	20.9
RPG	51.6	42.5	40.5	41.1
OTHER	37.3	31.1	35.9	35.9
PASCAL	0.0	6.1	9.7	9.7
TUTOR	1.0	2.6	2.6	2.5

TABLE 2

AVERAGE EXPENDITURES IN THOUSANDS OF DOLLARS
PER MINORITY OR TYPE INSTITUTION
COMPARED TO NON-M (NON-MINORITY) INSTITUTIONS
IN SAME CATEGORY OR TYPE FOR 1976 - 1977

AREAS OF ESTIMATED COMPUTER-RELATED EXPENDITURES	BY ENROLLMENT CATEGORIES						TYPE INSTITUTION (BELOW 10,000)				ALL INSTITUTIONS (BELOW 10,000)	
	BELOW 500		500 -- 2,499		2,500--9,999		PRIVATE		PUBLIC		REPORTING	
	MIN.	NON-M.	MIN.	NON-M.	MIN.	NON-M.	MIN.	NON-M.	MIN.	NON-M.	MIN.	NON-M.
HARDWARE PURCHASES	11	17	45	57	178	99	72	81	142	71	117	76
SOFTWARE PURCHASES	5	2	7	8	13	10	6	10	12	7	10	9
BUILDING & FURNITURE	--	6	10	13	40	12	27	9	25	14	25	12
COMPUTER STAFF	6	45	53	63	162	164	103	99	115	116	111	109
HARDWARE LEASE	--	36	26	39	137	107	95	67	87	77	89	73
SOFTWARE LEASE AND SERVICES	--	7	11	10	12	15	12	11	12	14	12	12
OTHER OPERATIONS	--	8	19	20	53	54	53	37	33	35	39	36
OFF-CAMPUS SERVICES	2	7	42	35	92	196	42	69	92	91	68	79

TABLE 3

AVERAGE EXPENDITURES IN THOUSANDS OF DOLLARS IN MINORITY AND NON-MINORITY INSTITUTIONS BY HIGHEST LEVEL OF OFFERING IN PUBLIC AND PRIVATE SECTORS
1976 -- 1977

GROUP	DEGREE	ENROLLMENT CATEGORY	AVERAGE (THOUSANDS) EXPENDITURE/INSTITUTION	
			MINORITY INSTITUTIONS	NON-MINORITY INSTITUTIONS
PUBLIC	ASSOCIATE	Below 500	-	28
		500- 2,499	56	76
		2,500- 9,999	537	194
		10,000-19,999	597	397
	BACHELOR	Below 500	-	-
		500- 2,499	160	119
		2,500- 9,999	105	344
	MASTER	Below 500	-	79
		500- 2,499	131	257
		2,500- 9,999	269	327
PRIVATE	ASSOCIATE	Below 500	-	121
		500- 2,499	55	42
		2,500- 9,999	-	207
	BACHELOR	Below 500	12	24
		500- 2,499	51	72
		2,500- 9,999	-	160
	MASTER	Below 500	-	57
		500- 2,499	255	83
		2,500- 9,999	202	244

TABLE 4

AVERAGE COMPUTER-RELATED EXPENDITURES PER REPORTING INSTITUTION BY FISCAL YEARS 1975 - 1977 AND FEDERAL INCOME SUPPORT TOWARD 1976 - 1977 COMPUTER BUDGET

(IN THOUSANDS OF DOLLARS)

CATEGORY	COMPUTER-RELATED EXPENDITURES			FEDERAL INCOME 76-77	PROJECTED EXPENDITURES 77-78
	74-75	75-76	76-77		
PRIVATE MINORITY INSTITUTIONS	$ 190	$ 200	$ 228	$ 63	$ 249
PUBLIC MINORITY INSTITUTIONS	182	211	304	52	273
ALL MINORITY INSTITUTIONS	185	207	277	58	265
ALL NON-MINORITY INSTITUTIONS	418	445	487	194	496
ALL REPORTING INSTITUTIONS	404	430	473	183	481

words, despite special funding available to minority institutions, these institutions were still not receiving funds for computing from federal sources in proportion to either their numbers or their needs.

Academic Computing

Collecting data on academic computing is difficult due to different methods of measurement, different things being measured, and the increasing lack of measurement (more trouble than it's worth) despite increasing emphasis on accountability in education generally. The data we need for accurate comparisons between minority and other institutions are not available as we go to press with this paper. However, we can probably safely make some assumptions on the basis of information already presented on hardware, software, and computer-acquisition dates. These boil down to one general observation: the differences in academic computing between the minority and non-minority institutions are largely a matter of scale and experience.

Table 5 shows faculty and student use in instruction and research in various disciplines in 115 minority institutions that reported data for 1976-1977. Costs per student varied among the disciplines, influenced by both the number of students and the type of use (whether intensive problem solving by individual students or simulation/demonstration-type projects involving many students at the same time).

As might be expected, schools granting degrees at the graduate level were the highest-spending groups in both the public and private sectors. However, nothing can be inferred from this about the quality of use. Public institutions spent nearly 80% of the total amount spent for academic computing in minority institutions. Almost as much was spent by the public two-year institutions alone as by all the private institutions together, probably because the two-year institutions enrolled more students.

The highest number of students using the computer were enrolled in computer science courses, engineering, business, and math and statistics, in that order. Education was lowest in use. The highest cost per student was in math and statistics, probably due to a high amount of individual problem solving. Business showed the lowest cost per student. This may have been due to a relatively high number of students and to a high proportion of simulation and demonstration-type applications. Other identifiable high-cost areas were psychology, engineering, and--to some extent--computer science.

Many of the costs per student can be expected to decrease in time as institutions gain experience and the computer is used in more courses and by more students. An additional factor that will enhance growth in academic computing on minority campuses is the development of suitable software. Many minority institutions, with heavy emphasis on remedial and tutorial programs for so-called high-risk students, are interested in courseware not usually heard of from the usual sources (such as CONDUIT or Conferences on Computing in the Undergraduate Curricula). Since resources for writing courseware (time, personnel, expertise, suitable software and hardware) are not as readily available in most minority institutions, it is clear there is no quick off-the-shelf solution to this problem (at least none conclusively proved to be both cost effective and educationally sound).

Some exceptions to the general pattern do exist, however, and some of these have been reported by HumRRO[4] as academic computing exemplars. One of these is a large minority time-sharing network, which provides services to other minority and non-minority institutions.

Although it is possible to question the cost effectiveness of academic computing in some of the disciplines as reported in Table 6, one must keep in mind such factors as length of experience, lower levels of funding (sometimes resulting in inappropriate hardware and software), and--looking to the future--the potential for capitalizing on less bureaucracy, coupled with experience in managing with less, and the recent emphasis on upgrading the management capability generally in minority institutions. Given time and resources (not necessarily at as high a level as in majority institutions), the minority institutions might make more cost-effective use of academic computing than others.

Charles Mosmann claims[5] that remedial education (an area with which most minority institutions identify because of their relatively high proportions of disadvantaged students) can be cost effective. Success in this area, however, does not come easily if neither the hardware nor the software (not to mention the courseware) is appropriate. Matching the real needs in the minority institutions with the institutions' strengths (as noted above, and not excluding their interest in teaching, rather than research) and with the appropriate resources or strategies to meet those needs is a challenge that must be met if the students served by these institutions are to graduate equipped to compete in many fields and thus become productive citizens to the extent of their potential.

Administrative Computing

Table 6 shows administrative applications most frequently reported as computerized and the percentage of computerized applications that were batch and on-line. The group most frequently computerized in minority institutions was that entitled Admissions and Records (in the FICHE survey) with Financial Management applications following closely behind. These parallel the trends in the total population. As was true for all institutions, Payroll was the most frequent financial application computerized.

All computerized applications in the

TABLE 5

EXPENDITURES FOR STUDENT AND FACULTY COMPUTER USE IN VARIOUS DISCIPLINES IN 115 REPORTING MINORITY INSTITUTIONS (INSTRUCTION AND RESEARCH) 1976 - 1977

NAME OF DISCIPLINE	AMOUNT SPENT	FACULTY USE	PERCENT OF FACULTY USE	STUDENT USE	PERCENT OF STUDENT USE	NUMBER OF COURSES	AVERAGE AMOUNT SPENT PER COURSE	NUMBER OF STU-DENTS	AVERAGE AMOUNT SPENT PER STUDENT
Engineering	$ 815,000	$163,000	20.00	$ 652,000	80.00	142	$4,591.55	4,939	$132.01
Envir. & Life Scs.	73,000	42,000	57.53	31,000	42.47	83	373.49	1,557	19.91
Computer Sciences	610,000	39,000	6.39	571,000	93.61	284	2,010.56	9,478	60.24
Math & Statistics	517,000	90,000	17.41	427,000	82.59	98	4,357.14	2,466	173.15
Physical Sciences	49,000	18,000	36.73	31,000	63.27	58	534.48	1,010	30.69
Psychology	28,000	3,000	10.71	25,000	89.29	33	757.58	175	142.86
Social Sciences	60,000	33,000	55.00	27,000	45.00	100	270.00	1,286	21.00
Education	3,000	2,000	66.67	1,000	33.33	8	125.00	58	17.24
Business	78,000	21,000	26.92	57,000	73.08	126	452.38	3,537	16.12
Other	320,000	65,000	20.31	255,000	79.69	74	3,445.95	1,772	143.91
TOTALS	$2,553,000*	$476,000	18.64	$2,077,000	81.36	1,006	$2,064.61	26,278	$ 79.04

---

* 79% of this amount was spent by public institutions. Of the 79%: 69% by graduate degree-granting schools
25% by associate degree-granting schools
6% by bachelor degree-granting schools

21% of this amount was spent by private institutions. Of the 21%: 49% by graduate degree-granting schools
32% by bachelor degree-granting schools
19% by associate degree-granting schools

TABLE 6

ADMINISTRATIVE APPLICATIONS MOST FREQUENTLY COMPUTERIZED IN MINORITY INSTITUTIONS 1976 -- 1977

				PERCENT OF COMPUTERIZED APPLICATIONS REPORTED AS:	
RANK	NO.	PERCENT	TYPE OF ADMINISTRATIVE APPLICATION	BATCH	ON-LINE
1	59	51.30	Class rosters[1]	91.53	8.47
1	59	51.30	Enrollment statistics[1]	94.92	5.08
2	58	50.43	Student registration process[1]	81.03	18.97
3	57	49.56	Term grade reporting[1]	92.98	7.02
3	57	49.56	Enrollment reporting[1]	98.25	1.75
4	55	47.83	Term student records and reports[1]	89.09	10.91
5	54	46.96	Course add/drop processing[1]	81.48	18.52
6	47	40.87	Undergraduate admissions process[1]	80.85	19.15
6	47	40.87	Payroll[2]	95.74	4.26
7	46	40.00	Departmental expenditures[2]	93.48	6.52
7	46	40.00	Financial aid accounting[2]	89.13	10.87
8	45	39.13	Student ethnic group reporting[1]	95.56	4.44
8	45	39.13	General fund ledger[2]	93.33	6.67
8	45	39.13	General fund expenditures[2]	93.33	6.67
9	44	38.26	Tuition and fee accounting[2]	86.36	13.64
10	41	35.65	Schedule of classes preparation[1]	78.05	21.95
10	41	35.65	Accounts payable	92.68	7.32

[1] Admissions and records

[2] Financial management

minority institutions were overwhelmingly reported to be batch operations, to an extent much greater than in the general population. Nearly a fourth reported no on-line use at all. More than 57% reported either no, or only one or two, on-line applications (and these were not major applications). In the established top ten applications, percentages of on-line use are quite small.

These data suggest that with regard to administrative applications, minority institutions are (a) following in the footsteps of majority institutions but (b) at an earlier point of developing, especially with regard to using state-of-the-art distributed data processing technology.

One might conjecture that in the future the gap may widen, the difficulty of upgrading to on-line systems being what it is. On the other hand, those minority institutions developing on-line applications today and in the near future may bear watching because they have had the opportunity to observe the successes and failures of the pioneers.

Computer Science Programs

Summaries of the FICHE survey data from the FICHE Form 3 show only two minority institutions offering a master degree program in computer science, 16 offering a bachelor, and 17 offering an associate degree during 1976-1977. None offered a doctorate. The total enrollment in the two master programs was only 58. All of these 58 students were enrolled at two historically black institutions; however they were not all black. Since the larger of the two master programs was established less than six years ago and many of the associate and bachelor programs were also recently begun, the computer science experience of minority institutions is considerably less than that of the total university group.

Minority institutions, by introducing programs in computer science later than the other institutions, were able to pattern their programs after the others. This results in a great similarity. The course offerings also show the influence of Association for Computing Machinery's recommendations.

The statistics show a need for additional computer science programs in the minority institutions. Only 5.4% of the associate degree programs reported were in minority institutions. Approximately 4.8% of the bachelor programs and 1.5% of the master programs are at minority institutions. No minority institution offers the doctorate degree in a computer-related field.

Table 7 shows the number and percent of programs, majors, and degrees at minority institutions by level of institution. Minority schools reported an enrollment of 3,805 undergraduate majors in computer science and related programs. This represents 6.4% of the total for all institutions. The minority programs awarded 405 undergraduate degrees representing 4.8% of the total.

There are some staffing difficulties. It is to be expected that with doctorates in computer science at a premium nationally, those institutions with less financial resources would find it difficult to attract qualified personnel to staff computer science programs. The minority institutions reported a total staff of 13 with doctorates in computer science (see Table 8), or 0.42 computer science doctorates per program. Overall there were 41 faculty and staff with doctorates, or 1.32 per program. This significantly low percentage of qualified staff in computer science programs may point to a rocky road ahead for computer science programs in minority institutions, particularly at a time when retrenchment in numbers of majors is being called for by boards of trustees, accrediting agencies, and even federal agencies, despite the well-documented national manpower needs.

Staffing

Not only do computer science programs require qualified personnel. So also do computer centers. Table 9 shows the distribution of professional staff by employment categories for minority and non-minority institutions. Percentages of expenditures for professional staff were a little higher in non-minority institutions in 1976-1977. It is not surprising to find higher FTE staff percentages in such categories as analyst and systems programmer in non-minority institutions. The need for them is more clearly seen by experienced institutions. On the other hand minority institutions spent more of their staff budgets on applications programmers (probably more in administrative than academic computing). This should be expected if these installations are in their start-up years. However, in some instances, where such machines as IBM 1130s have been in place for a number of years, the term application programmer may represent little in the way of new development; it may represent a high percentage of non-productive maintenance programming. These differences in minority and non-minority institutions do not point to any significant conclusions.

Table 9B also shows few differences. The only noticeable difference is in the percentage of minority institutions with professional personnel with a doctorate in computer science (.31%), as compared with the corresponding percentage in majority institutions (.78%), although in both cases it is under one percent. This suggests possibly more emphasis in the minority institutions on administrative, rather than academic, computing. Computer centers in majority institutions traditionally originated with National Science Foundation grants in the sixties, to which minority institutions did not have access. NSF requested that the computers thus acquired be used for academic purposes, and in many cases it took administrators several years to discover the potential and

TABLE 7

LEVEL OF INST.	LEVEL OF DEGREE	NUMBERS OF COMPUTER SCIENCE PROGRAMS, MAJORS, AND DEGREES AWARDED IN 1976-1977 BY MINORITY INSTITUTIONS REPORTING IN FICHE SURVEY (IN PARENTHESES: PERCENTS AS COMPARED TO ALL INSTITUTIONS IN THE SAME CATEGORY)											
		COMPUTER SCIENCE			DATA PROCESSING			PROGRAMMING AND INFORMATION SYSTEMS*			OTHER		
		PROGS.	MAJORS	DEGREES	PROGS.	MAJORS	DEGREES	PROGS.	MAJORS	DEGREES	PROGS.	MAJORS	DEGREES
ASSOCIATE	ASSOCIATE	0 (0)	0 (0)	0 (0)	9 (5.7)	1672 (9.1)	183 (6.4)	2 (6.4)	100 (2.3)	49 (8.6)	1 (3.5)	245 (9.7)	19 (5.8)
BACHELOR	ASSOCIATE	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	1 (17.4)	340 (38.1)	31 (43.1)
	BACHELOR	0 (0)		0 (0)	0 (0)		0 (0)	0 (0)		0 (0)	2 (12.4)		7 (4.0)
MASTER	ASSOCIATE	3 (18.9)	1067 (13.9)	8 (10.7)	1 (10.0)	216 (10.0)	12 (19.4)	1 (34.0)	190 (21.9)	0 (0)	0 (0)	104 (11.6)	0 (0)
	BACHELOR	9 (12.1)		63 (5.4)	1 (5.9)		0 (0)	2 (20.0)		27 (19.2)	2 (6.3)		9 (5.2)
	MASTER	2 (15.5)	58 (7.5)	9 (7.8)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)

* (at master degree-granting institutions)

TABLE 8

FACULTY
IN COMPUTER SCIENCE PROGRAMS
AT MINORITY INSTITUTIONS
REPORTING IN FICHE SURVEY
1976 -- 1977

DEGREE AREA OF FACULTY	NUMBER	% OF TOTAL
Doctorate in Computer Science (FT)	13*	5.0
Other doctorates (FT)	29*	11.0
Other full-time non-doctorates	81*	30.8
Teaching and research assistants (PT)	56**	21.3
Other part-time staff	82**	31.2

* 123 = total full time
** 138 = total part time

184 = full-time equivalent (FTE)

TABLE 9

PERCENTAGE DISTRIBUTION OF PROFESSIONAL STAFF

A. BY EMPLOYMENT CATEGORIES*

CATEGORY	NO. OF FTE STAFF	% OF 1976-1977 EXPENDITURES USED FOR STAFF	FTE STAFF PERCENTAGES (of total staff budget)					
			MANAGER	CLERK	ANALYST	SYSTEMS PROGRAMMER	APPLICATION PROGRAMMER	OPERATOR
ALL MINORITY INSTS.**	652	36.16	16.72	14.42	8.28	5.52	20.00	34.97
NON-MIN. INSTS.** 10,000	7,192	39.87	17.24	14.57	10.19	6.02	17.54	34.43

B. BY HIGHEST LEVEL OF EDUCATION*

CATEGORY	NO. OF STAFF	DOCTORATE			MASTER			BACHELOR			UNDER-GRAD.	ASSO-CIATE	H. S. DIPLOMA
		NO. IN COMPUTER SCIENCE	NO. OF OTHER	PER CENT	NO. IN COMPUTER SCIENCE	NO. OF OTHER	PER CENT	NO. IN COMPUTER SCIENCE	NO. OF OTHER	PER CENT	PER CENT	PER CENT	PER CENT
ALL MINORITY INSTS.**	635	2	24	4.09	21	72	14.64	27	126	24.09	12.12	9.60	35.43
NON-MIN. INSTS.** 10,000	7,823	61	277	3.99	233	664	10.28	348	1,538	22.16	17.26	9.16	36.76

* 1976-1977
** reporting

begin to use the resource. The only competitor to student use was large faculty research projects. Computer center directors often had joint academic appointments in the early computer science programs.

This establishment was long a closed world to minority institutions, and the programs offering such grants (specifically targeted toward computing hardware acquisition and its support) dried up before most minority institutions were aware they existed or were able to take advantage of them. In other words, academic computing in minority institutions did not benefit from the funds available to give academic computing a start. There may be a connection between this historic situation and the difference in the percentage of professional staff in minority and majority institutions with a doctorate in computer science. However, this gap is relatively unimportant except insofar as it draws our attention to historical inequities (more on this later) directly related to the development of computing in minority and non-minority institutions.

Historical Factors

Most of the data cited in this study give the reader a view of one slice in time, the year 1976-1977 (with enrollment data for fall, 1975). In studying a group of institutions whose histories are known to have differed strikingly from those of the larger group (the non-minority institutions), it is important to bring background information to bear on the data as they are interpreted. Therefore, before summarizing and making general conclusions, some historical facts are briefly considered here.

Computing in an educational institution supports the goals of that institution to the extent that it is seen as a priority supporting other priorities and to the extent that resources are available. Minority institutions in general did not look upon computing as something they should pursue until the late sixties for the following reasons (and if they did, few could muster the resources).

Scores of years of forced second-class status, plus the specter of being put out of business by desegregation, had caused minority colleges to begin to question, during the late fifties, whether they could or should exist. Not until 1964 did the gloom start to dissipate, largely as the result of the McGrath Report (by Earl McGrath, a scholarly reaffirmation of the need for continued existence of the black schools). Financial aid legislation, Title III programs, and a variety of other programs followed; and the prospects of proving their worth and realizing their potential loomed before the minority schools like pillars of fire. For some it was too little too late, but for others these and related events enabled the institutions to rekindle their sense of purpose. The civil rights struggle in the sixties for blacks and the American Indian meetings begun in the early sixties also helped to revitalize the feeling that each of these minorities had at least as much to offer to the majority society as the larger society had to offer them. Motivation was high. As the gates of Washington swung open, the National Science Foundation was among the agencies discovered. A small percentage of NSF funds became available to the minority institutions to improve their science programs in the early seventies. (Minority institutions included only the historically black schools at first, but soon the definition was broadened to include other minorities and two-year institutions. Funding, however, was not increased in the same proportion.)

As the institutions began to advance toward generally higher levels of sophistication in management, granstmanship, and accountability, computing began to move from back-burner to priority status. All this happened in very recent times, and it is too early to expect widespread comparability to the majority institutions. The fact that some minority institutions have already found their way into the mainstream of academic computing exemplars as reported by HumRRO[4] suggests that with appropriate resources minority institutions can make appropriate contributions. The reader of this paper should probe further than the mere facts reported here. A sense of history and its impact on computing can help to explain the differences that have been found in computing in minority and non-minority institutions.

Summary and Conclusions

The minority institutions, as shown by data examined in this study, appear to be following--in general--the lead of the majority institutions with a view toward catching up. However there is no evidence that the gap has narrowed yet. Some indicators point to more trouble ahead, while others point to possible significant progress.

Academic computing in minority institutions appears to contain some trouble spots with regard to both the present and the future. The 1976-1977 data suggest that minority institutions were not gaining experience with state-of-the-art micro computer software or with graphics. Hardware acquisition is a recent development in most of the institutions, based on the higher expenditures for equipment, buildings, and furniture that were computer related. Beginnings have been made, but more time is needed and more expertise.

Administrative computing shows a similar situation, with a high proportion of batch applications and few on-line applications. However, with the current emphasis on improving the planning, management, and evaluation capabilities of minority institutions, it appears that those institutions currently developing on-line applications have the potential (if the hardware is appropriate) to be highly successful because enough models exist in the majority institutions to make it unnecessary to reinvent the entire wheel.

The largest gap in computing between minority and majority schools appears to exist in computer science programs. Only two at the master level exist on minority campuses. Despite national manpower needs, computer science programs tend to face a negative political climate.

Minority institutions seem to be increasing in number, at least among American Indians and Spanish-speaking people, but the newest institutions are not yet accredited.

Funding for minority institutions for computer-related expenditures has been, on the average, considerably less than for other institutions. When funds have been made available for this purpose, more institutions have subsequently been given the opportunity to compete, and the funding has not increased proportionately.

The rate of increase in spending for computers has gone up faster in recent years for minority than non-minority institutions, suggesting that minority institutions are beginning to see computing as a priority more than in the past.

Not enough information is available on whether computers can be cost effective in educating minority students, but some factors seem to give cause for cautious optimism. Use of computers in remedial and tutorial work matches an area in which minority institutions are extremely interested because of their commitment to large numbers of high-risk students. This is an area in which computers have proved cost effective and to some extent educationally effective. However, appropriate hardware, software and courseware are necessary, and these do not tend to exist yet on many minority campuses, nor is much funding available from any source specifically for this purpose. It is not yet clear that networks can answer the problem because of high communication costs in some cases and lack of readiness in others.

Additional potentials for cost effectiveness exist in minority institutions in the form of experience in managing with meager resources, the general lack of entrenched bureaucracies, and current emphasis on improving the ability to plan, manage, and evaluate.

The general conclusions one can make, based on the FICHE data considered in this study and historical considerations, can be summarized as follows:

(1) Minority institutions do not differ radically from majority institutions in what they are attempting to do with computers, but they are at an earlier point on the continuum of development (with few exceptions) due to later beginnings.

(2) The type of hardware and software existing in most of the minority institutions does not point to an early narrowing of the gap between them and the nonminority institutions.

(3) The minority institutions in general have the potential to advance, even significantly in some cases, given the time and resources.

Footnotes

1 FICHE: Fourth Inventory of Computers in Higher Education, supported by the National Science Foundation under grant SED 75 17157 to the Computer Science Department of the University of Missouri-Rolla under the direction of John Hamlen

2 Needs/Strategy Evaluation of Minority Institutions in Educational Computing, in progress and supported by the National Science Foundation under grant SPI7821515 to Xavier University under the direction of Sr. Patricia Marshall

3 Based on data from U. S. Office of Education (enrollment data as of fall 1975)

4 HumRRO (Human Resources Research Organization): Academic Computing Directory, First Edition (Alexandria, 1977).

5 Mosmann, Charles. Evaluating Instructional Computing, pages 24-28 (University of California, Irvine, 1976). Research funded by National Science Foundation Grant TIE-EPP74-12816.

Trends in Technology for Educational Computing

Chaired by Harry Hedges
Michigan State University

Microcomputer Developments and Their Relation to Education

Kenneth Bowles, University of California-San Diego

As microcomputers become increasingly available and attractive for use in individual departments within a college or university, faculty and administrators are more and more involved in planning uses to which these computers can be put and criteria for their evaluation. This presentation will focus on current microcomputer developments, their uses in education, selection of particular units, selection of a language or languages to be used, and implementation of those selections.

The Role of Videodisc Technology in Education

Leo Leveridge, American Medical Association

The advent of industrial-type optical videodisc systems has brought us to the exciting beginning of a revolution in communication potentially as significant to education as the invention of movable type some 500 years ago. Although computers have enabled important educational advances in recent decades by automating the communication of words and, with branching, mading verbal materials interactive, these data processors have done little to transmit audiovisual data because, until the arrival of the videodisc, it has not been practical to teach audiovisually by computer. Conversely, a serious lack of audiovisual education is that heretofore it has not been interactive. Now, however, thanks to the configuration of optical videodisc technology, it becomes feasible and economical to integrate both forms of communication into powerful systems for more complete self-instruction.

The special advantages of videodisc players for Interactive Multimedial Tailored Self-Instruction with Self-Assessment will be described in the presentation. A closed-circuit, in-room demonstration of the Thomson-CSF prototype industrial optical videodisc player will be held in cooperation with the Computer Assisted Instruction Laboratory and the Department of Instructional Television of the University of Iowa.

Future Trends in Computer Technology with Implications for Education

Samuel Patton, Rensselaer Polytechnic Institute and IBM Data Systems Division Laboratory

Future and current circuit technology, as it has evolved from relays to LSI, used in computing systems and the evolution of computer technology will be discussed. Then, a look at trends in technology and architecture for the future and implications for the uses of computers in education will be presented.

The Electronic Poet--Color Graphics Poetry: A Vision of Information

Edmund Skellings
Florida International University

Abstract not available at time of printing.

QUANTUM CHEMISTRY INSTRUCTIONAL PACKAGE USING INTERACTIVE FORTRAN[†]

Bhairav D. Joshi*, Department of Chemistry, State University College, Geneseo, NY 14454
James E. Eilers, Department of Chemistry, State University College, Brockport, NY 14420

This paper deals with the Quantum Chemistry Experimental System (QCES) designed primarily for use by undergraduates in physical chemistry and introductory quantum chemistry courses. Computer-assisted instruction has become a valuable tool for teaching chemistry at various levels of undergraduate curricula, particularly in physical chemistry. Computer-assisted experiments have also been tried. These two ideas have been further exploited in developing the QCES package. This system of interactive computer programs is expected to be a valuable pedagogic tool in helping students get a better understanding of the concepts and techniques of quantum chemistry.

I. INTRODUCTION

The subject of quantum chemistry has become very important in the training of chemists, even at the undergraduate level. The study of quantum theory is no longer a senior level elective course, but has become a most important part of the standard undergraduate course in physical chemistry.[1,2] Some teachers have chosen to introduce qualitative concepts of quantum mechanics to freshman chemistry majors,[3] and even to general audiences of mixed backgrounds and career goals.[4] Quantum chemistry is, however, by its very nature, a highly mathematical subject; students somewhat deficient in mathematical sophistication often get so bogged down in or turned off by the mathematical details that they never recognize the fundamental principles or the more qualitative concepts of quantum theory. Even for the more mathematically oriented student, appreciation of the consequences of quantitative quantum chemistry is often inhibited by the complex and time consuming calculations necessary to investigate the effects of the numerous variables. Furthermore, even for the simple model systems with which we, the teachers of quantum chemistry, tend to introduce and illustrate the concepts and techniques, the complete mathematical analysis is a formidable task that is of questionable value for beginners or for non-physical chemists. Nevertheless, it is preferable that the student learn the qualitative aspects by discovery rather than by blind acceptance and that in the early learning stages be, if at all possible, somewhat protected from his/her own mathematical ineptitude.

It is our belief that with the widespread availability of real time computing systems, it is now appropriate to construct novel types of computer experiments that can circumvent these obstacles. Such experiments can be designed to help the student to focus on concepts while the computer does the mathematics in a quick and error-free manner behind the scenes. This efficiency will allow the better student to investigate many more remifications that was traditionally possible, and also allow the error-prone student to examine the consequences of correct quantitative calculation.

Several isolated attempts have been made along these lines by teachers of quantum chemistry. Frankel and Davis[5] proposed an experiment for calculating the ionization potential of closed-shell atoms using the Hartree-Fock-Roothaan Self-Consistent-Field theory.[6] Janis and Peterson,[7] and Murgich[8] have created a quantum chemical experimental system, called UNIQUE, to study a π-electronic structure of molecules using the Hückel theory.[9] Much work has been done on the computer-aided determination[10-12] of the Russell Saunders spectroscopic terms. The pedagogic value of computer-generated charge densities and molecular orbitals in understanding the formation of chemical bonds has been well documented by Wahl.[13]

A number of excellent computer-oriented experiments dealing with specific systems, rather than a particular concept, have also been published recently. Dewar and Klemen[14] illustrate the many facets of LCAO-MO theory in the treatment of the H_2 molecule and its simplest-of-all covalent bond. Snow and Bills[15] suggest a LCAO-MO experiment based on He. Robiette[16] treats the simplest one-electron bond in H_2^+ using the Variational Method first discussed by Coulson.[17] Lapidus[18] has suggested an interesting one-dimensional model for a two-electron system. This model illustrates approximation methods while avoiding complex mathematical analysis.

In an interesting, recent article Bork[19] discussed the ways computers can serve as aids to increasing physical intuition. Dowd[20] has published an account of how computers can be used in a laboratory as "laboratory assistants." An account of the use of computerized visual aids in introducing the molecular orbital theory has recently been published by McGrath et al.[21] Finally, Krass[22] has shown how an interactive computer program can be used to simulate quantum measurements on a one-dimensional, simple, harmonic oscillator.

In spite of these attempts, there has been little systematic effort devoted to the needs of an average student just beginning his study of quantum chemistry. Experimental systems are needed to help the beginner appreciate the many and varied consequences of quantum mechanical principles as they are developed in his/her courses. When coupled with the traditional CAI techniques, these experiments can provide the nucleus of a self-paced learning experience.

For the last few years we have been engaged in developing a Quantum Chemistry Experimental System (QCES) to be used primarily by undergraduates and beginning graduate students of quantum chemistry. Our pedagogic objective has been to circumvent the barrier of complex mathematics using interactive computer programs. We want the student to concentrate on the physical aspects of the quantum mechanical principles at this stage of development. Whenever possible we illustrate the concepts and techniques using simple model systems (a particle in a box, a rigid rotor, or a harmonic oscillator) of quantum chemistry.

As used by the student, the QCES appears to be organized in terms of concepts and techniques. (S)he decides at the beginning of a given study-session to learn about a topic such as the Variational Principle or Perturbation Theory. The programs themselves are organized as much as possible around the model systems, i.e., a particle in a box, and methodology.

II. THE PARTICLE-IN-A-BOX (PIB) SYSTEM

Once the basic postulates of quantum theory have been introduced, most quantum chemistry courses proceed with a fairly detailed examination of a particle in a simple one dimensional box.[23-24] Study of this simple system and its straightforward generalizations (finite barriers, two- and three-dimensional boxes, etc.) can help a student appreciate many concepts that also apply to more realistic systems. Some of such general features, illustrated by the PIB problem are:

(a) quantization of energy, and line spectra,[25]
(b) degeneracy, quantum numbers, and boundary conditions,[25]
(c) tunnelling through potential barriers,[26-28]
(d) the DeBroglie relationship,[25]
(e) the expansion theorem,[29a]
(f) the variational principle,[29b]
(g) simple perturbation theory,[25] etc.

In addition, the understanding of the PIB problem exposes the student to other fundamental concepts and terminology used in quantum mechanics, i.e., normal, linear, and Hermitian operators; eigenvalues and eigenfunctions; symmetry properties of wave functions; expectation values of operators; constants of motion; stationary states, etc.

The basic goals of the PIB Experiments are to provide the student an easily accessed and easily used set of programs -- the PIB system, with which he can quickly and accurately generate data concerning the bound states of particles in one-dimensional square wells of varying depths and widths, and their two- and three-dimensional generalizations. The student is expected to use these data to undertake an indepth exploration of the several characteristics of these solutions. This exploration is guided by appropriate computer generated prompts to the student.

The PIB system is designed to have the following general features. First, it is interactive. Second, it requires no prior computing or programming experience by the student. Third, it is able to generate graphs and plots of the numerical data. Finally, the system provides hints if the student indicates difficulty with the preprogrammed problems and questions. It is also highly desirable that the system keep track of the progress of both the individual student as well as the group as a whole while providing the instructor with a record of student responses. The whole network of routines should be tied together with an elaborate branching and loop-back system that optimizes students' routes according to their responses (see Sec. V).

Use of the PIB experiment should enhance student understanding of the following important quantitative concepts:

(a) the dependence of energy level spacings on the mass of the particle, the well-width, and the height of the potential well,
(b) the dependence of the particle's deBroglie wavelength upon the particle's mass, on the width of the potential well, and on the depth of the well,
(c) the finite probability of a particle tunnelling through finite potential barriers,
(d) the expectation values of p_{op}, p^2_{op}, x_{op}, x^2_{op}, etc., and their relation to the Heisenberg's uncertainty principle,
(e) the zero-point energy and the uncertainty principle,

(f) the use of perturbation theory in solving the Shrödinger equation, and the nature of perturbation corrections to both the wave functions and energies,
(g) families of orthonormal functions, and expansion of any arbitrary function in terms of a complete set of orthonormal functions -- convergence and other practical aspects, i.e., truncation of such expansions,
(h) emission and absorption spectra and their relation to energy levels, occupation numbers, selection rules, symmetry, etc.

III. A LIST OF COMPUTER-ASSISTED EXPERIMENTS BEING DEVELOPED

1. Particles in boxes: one-, two-, and three-dimensional boxes with varying potential depths and shapes.
2. Oscillators (harmonic and anharmonic): interpretation of vibrational spectra of simple diatomics.
3. Rigid and simple non-rigid rotors: interpretation of vibronic spectra of diatomic molecules.
4. Approximation methods (variational and perturbative techniques): applications to model systems as well as to a few real systems.
5. The H-atom, the independent particle model; central-field potentials; the periodic properties of atoms.
6. The LCAO-MO theory for He, H_2^+, and H_2.
7. Angular momentum, ladder operators, spherical harmonics, term symbols, and degeneracies.
8. Orbitals: types, hybridization; spatially oriented orbitals.
9. Hückel theory: application and interpretation of molecular structure.
10. Hartree- Hartree-Fock theory of atoms; an illustrative application to many electron atoms.
11. Symmetry, group theory, and molecular structure.
12. Drills (typical examples are listed below):
    a) Mathematical techniques: Orthonormalization, matrix diagonalization, secular equations, iterative methods, properties of operators, coordinate transformations, vectors, similarity transformations, generating functions, and recursion relations, etc.
    b) Concepts in quantum chemistry: quantum numbers, term symbols, electronic configurations of atoms and molecules, ladder operators, projection operators, symmetry and group theory, character tables and properties of orbitals, selections rules, correlation diagrams, concepts leading to the formation of quantum theory (e.g., black body radiation, photoelectric effect, the Bohr atom, etc.), Hund's rules, Pauli principle and antisymmetry, Slater determinant, etc.

IV. STRUCTURE OF QCES

Considerable progress has been made towards achieving the goals set forth in Sec. I. Our preliminary activities concentrated on

(a) preparing comprehensive plans for approaching and refining the problem, and
(b) conceiving, developing, and testing the several levels of interactive and non-interactive computer programs needed to attain the desired objectives.

Our plans call for four fundamental levels of programs:

LEVEL I programs: Service routines (i.e., plot a function, print a matrix, find area under the curve, etc.)

LEVEL II programs: Limited and specific task routines (i.e., given l, m, θ, and ϕ, calculate the real spherical harmonics.)

LEVEL III programs: Routines which interact with the student in order to teach him/her something. These routines are to include (a) simple drill programs and exercise setters, (b) limited purpose drivers that enable students to call upon lower level programs to do things such as plotting electron density contours of a 3d(xy) orbital in the xy-plane, and (c) lesson programs that call upon lower level programs so as to tailor make a lesson for each student.

LEVEL IV programs: These are to be monitor and control routines to assist the instructor in evaluating the programs, class progress, and class feedback.

A list of programs developed and tested thus far is given in Appendix (A).

Another idea that needs to be exploited further for inclusion in our programs package is the use of FORTRAN in the Calculator Mode: During interactive sessions on a terminal the programmed lessons will often ask that the student give a numerical answer to a short problem. When this happens the student should be able to shift the terminal to calculator mode and use it (in much the same manner as a hand held calculator) to arrive at the desired numerical answer. (This is fairly common in most interactive languages[30,31], but not previously in FORTRAN. We have developed a way of doing it in FORTRAN on the PRIME computer and are currently developing the procedure.) Having arrived at a numerical result the student may then return to the lesson at the exact point of departure and continue the interactive session.

Plans are underway for class-room testing of all programs, and the results will be reported in a forthcoming paper.

V. CONCLUSION -- USEFULNESS OF THE QCES PACKAGE

The pedagogic value of CAI is amply demonstrated by the wealth of published literature on this subject.[32] Computer-assisted experiments have also been successfully used.[3-18] Our plans envision a synthesis of these two approaches in creating the QCES package for use in an introductory course in quantum chemistry. In using the QCES package the student is to be encouraged to examine concepts, trends, and methods and leave the actual computations to the computer. Such an approach has three major advantages:

(1) The student examines many facets of a series of problems in quantum chemistry in a reasonable amount of time.
(2) All computational mistakes by the student and major mathematical barriers are removed, thus paving a way for the student to concentrate on acquiring a thorough quantitative understanding of the quantum chemical concepts and techniques.
(3) The student will hopefully feel the excitement of discovery and develop his analytic faculties rather than simply accepting wisdom from the teacher.

As a result of extensive interaction with the QCES package, the student should:

(a) find the subject matter of quantum chemistry far less intimidating,
(b) get a better return for the time he invests in the study of quantum chemistry,
(c) find that he does not get lost or draw erroneous conclusions due to his having made a mathematical mistake, and
(d) be aided by fine computer graphics facilities in the visualization (or graphical interpretation) of the many complex, mathematical formulations that relate physical parameters.

Therefore, the student should:

(a) gain greater insight into the fundamental principles and their consequences,
(b) find it stimulating, yet convenient, to investigate the effect that varying various physical parameters have on the properties of a given system,
(c) be able to discover for him/herself the salient features, theories, and interpretations that so often fail to have the desired lasting impact when encountered in lectures or text,
(d) develop or hone analytical skills,
(e) and find the study of quantum chemistry to be a more enjoyable and stimulating experience.

The instructor should find the QCES package to be an exceptionally effective and tireless Teaching Assistant that is always available to any student upon demand. Consequently, classes can cover more material, and classroom efforts can concern themselves with aspects that could not ordinarily receive enough attention. Moreover, there should be several beneficial carryover effects such as an improved knowledge base for courses with physical chemistry prerequisites, improved student attitude and morale paralleling their greater success in physical chemistry, and improved enrollment in junior-senior level courses as the fearsome reputation of physical chemistry erodes.

VI. REFERENCES AND FOOTNOTES

†Supported by a grant from the Research Foundation of the State University of New York; Grant #125-4006A.

*Author, who will deliver the paper, and to whom all correspondence should be directed. Telephone # 716-245-5305.

1. R. M. Rosenberg, Principles of Physical Chemistry, Oxford Univ. Press, New York (1977).
2. W. J. Moore, Physical Chemistry, 4th ed., Prentice-Hall, Inc., Englewood Cliffs, N.J. (1972).
3. D. G. Barnes, J. Chem. Ed., 51, 396 (1974).
4. The Science Foundation Course Team, An Approach to the Teaching of Quantum Theory, in "Wave Mechanics -- The First Fifty Years," W. C. Price, S. S. Chissick, and T. Ravensdale, eds., Halsted Press, New York (1973).
5. E. C. Frankel and D. D. Davis, J. Chem. Ed., 50, 80 (1973).
6. C. C. J. Roothaan, Rev. Mod. Phys., 23, 69 (1951).
7. F. T. Janis and E. J. Peterson, J. Chem. Ed., 50, 622 (1973).
8. J. Murgich, J. Chem. Ed., 54, 421 (1977).
9. E. Hückel, Z. Physik, 70, 204 (1931).
10. B. D. Joshi, J. Chem. Ed., 53, 245 (1976).
11. J. H. Wise, J. Chem. Ed., 53, 497 (1976).
12. K. E. Hyde, J. Chem. Ed., 52, 497 (1976).
13. A. C. Wahl, Science, 151, 961 (1966).
14. M. J. S. Dewar and J. Keleman, J. Chem. Ed., 48, 495 (1971).
15. R. L. Snow and J. L. Bills, J. Chem. Ed., 52, 507 (1975).
16. A. G. Robiette, J. Chem. Ed., 52, 95 (1975).
17. C. A. Coulson, Trans., Faraday Soc., 33, 1479 (1937).
18. I. R. Lapidus, Am. J. Phys., 43, 790 (1975).
19. A. Bork, Am. J. Phys., 46, 796 (1978).

20. J. Dowd, Am. J. Phys., 46, 63 (1978).

21. T. McGrath, M. Kroeger, and H. Dunn, J. Chem. Ed., 55, 160 (1978).

22. A. K. Krass, Am. J. Phys., 46, 152 (1978).

23. I. N. Levine, Quantum Chemistry, Allyn & Bacon, Inc., Boston, MA (1974).

24. A. R. Denaro, A Foundation for Quantum Chemistry, John Wiley and Sons, New York (1975).

25. M. W. Hanna, Quantum Mechanics in Chemistry, W. A. Benjamin, Inc., New York, 1969, pp. 51-72.

26. See ref. 24, Ch. 4.

27. P. A. Deutschman, Am. J. Phys., 39, 952 (1971).

28. M. Campi and M. Harrison, Am. J. Phys., 35, 133 (1967).

29. D. V. George, Principles of Quantum Chemistry, Pergamon Press, New York (1972). (a) pp. 46-47. (b) pp. 106-108. Also see problem #7-3.

30. For the "calculator" mode in APL see: (a) A. D. Falkoff and K. E. Iverson, The Design of APL, IBM J. Res. and Dev. 17, 324 (1973), and (b) R. P. Polivka and S. Pakin, APL - The Language and Its Usage, Prentice-Hall, New York (1975).

31. For the "calculator" mode in FOCAL and BASIC see, for example, PROGRAMMING LANGUAGES, Digital Equipment Corporation, 1970.

32. J. S. Mattson, H. B. Mark, Jr., and H. C. McDonald, Jr., eds., Computer Assisted Instruction in Chemistry. Part (A): General Approach, Part (B): Applications, Marcel Dekker, Inc., New York (1974).

APPENDIX (A)

This appendix presents an alphabetical listing of all programs developed and tested thus far. The following abbreviations are used here:

RR: Rigid rotor
PIB: Particle in a box
SHO: Simple harmonic oscillator

#	PROGRAM	DESCRIPTION
1	ANGMOM	Calculates the angular momentum of a RR.
2	CALC	Used by EXPAND to calculate various approximations to e^t in terms of PIB wave functions using the quantum mechanical expansion theorem.
3	CLPROB	Calculates SHO's probability of being in the classically forbidden region.
4	CONPOL	Applies the PIB model to conjugated polyenes.
5	CONTOUR	Plots two-dimensional electron-density contours of atomic orbitals.
6	COSPOT	Calculates and stores the first-order perturbed wave function of a PIB with cosine bottom.
7	COSPRT	Applies first order perturbation theory to the problem of a PIB with a cosine bottom.
8	DFX(E)	Evaluates the first derivative of energy of a PIB with finite depth.
9	EHO	Calculates the energy of a SHO.
10	ENERGY	Calculates the energy of a RR.
11	EPIB	Calculates the energy of a PIB.
12	EPOTWL	Calculates and tabulates the energy of a PIB with finite depth.
13	ETRANS	Calculates energies of allowed rotational transitions.
14	EXPAND	A graphical illustration of the quantum mechanical expansion theorem.
15	FAL	Calculates unnormalized associated legendre functions.
16	FL	Calculates unnormalized legendre functions.
17	FPSI	Calculates the PIB wavefunction.
18	FREQ	Calculates the fundamental vibrational frequency of a diatomic molecule given the force constant.
19	FX(E)	Evaluates the energy of a PIB with finite depth.
20	GOCLPROB	Interactive 'driver' for CLPROB.
21	GOCOSPOT	Interactive 'driver' for COSPOT.
22	GOCOSPRT	Interactive 'driver' for COSPRT.
23	GOEPOTWL	Interactive 'driver' for EPOTWL.
24	GOFAL	Interactive 'driver' for FAL.
25	GOGRAPH	Interactive 'driver' for GRAPH.
26	GOHERM	Interactive 'driver' for HERM.
27	GOPERT	Interactive 'driver' for PERT.
28	GOPOLYENE	Interactive 'driver' for CONPOL.
29	GOROTPROB	Interactive 'driver' for calculating properties of a RR.

#	PROBLEM	DESCRIPTION
30	GORSH	Interactive 'driver' for RSH.
31	GOSIMP	Interactive 'driver' for SIMP.
32	GRAPH	Plots F(X) vs X, given the values.
33	HERM	Calculates Hermite polynomials.
34	HNK	Calculates the coefficients which determine the first order wave function for a PIB with sloping bottom.
35	HODIS	Calculates relative populations of states of a RR.
36	HODIST	Interactive 'driver' for HODIS.
37	HOOKE	Calculates the force constants for a diatomic molecule using the SHO approximation.
38	HYDRO	Generates tables and plots of radial hydrogenic wave functions and their squares.
39	P2BAR	Calculates the value of $\langle p^2 \rangle$ for a PIB.
40	PERT	Applies the first order perturbation theory to a PIB with sloping bottom.
41	PIBOO1	This is an interactive program dealing with various aspects of quantum mechanical principles and uses the numerous PIB programs to illustrate them.
42	PIBFNS	Interactive 'driver' for calculating the properties of a PIB.
43	PIBFPLOT	Calculates and plots $\psi_n(x)$ and $\psi_n(x)^2$ for a PIB.
44	PIBNU	Calculates the frequency of the radiation emitted by a PIB during a transition: $n_1 \rightarrow n_2$.
45	PIBSTA	Keeps statistical record of a student's performance during a given interactive session.
46	PSHIO	Calculates the wave function for a SHO.
47	PUNCER	Calculates $\langle \Delta p \rangle$ for a PIB.
48	R	Calculates the bond-distance for a diatomic molecule using the RR approximation.
49	R12	Calculates and tabulates the atomic two electron integrals over hydrogenic functions.
50	RELDIS	Calculates the relative population of states for a RR.
51	RHOTAU	Calculates rotational frequency and the period of rotation of a RR.
52	RRPROGS	Calculates properties of a RR.
53	RSH	Calculates the real spherical harmonics.
54	STATES6	Generates Russel-Saunder's atomic term-symbols.
55	THYDRO	Calculates Laguerre polynomials.
56	TRADIA	Calculates the radial hydrogenic wave functions.
57	XBAR	Calculates $\langle x \rangle$ for a PIB.
58	X2BAR	Calculates $\langle x^2 \rangle$ for a PIB.
59	XINUM	Generates and tabulates the integrals $\langle \psi_m(x) \vert x^\nu \vert \psi_n(x) \rangle = XI(\nu,m,n)$ for SHO wave functions.
60	XUNCER	Calculates $\langle \Delta x \rangle$ for a PIB.

FIVE APPLICATIONS OF COMPUTERS IN CHEMICAL EDUCATION

K. Jeffrey Johnson
Department of Chemistry
University of Pittsburgh
Pittsburgh, PA 15260

INTRODUCTION

Five applications of computers in chemical education have been developed during the past decade at the University of Pittsburgh. These applications include:

1. A library of 45 computer-assisted instruction (CAI) lessons and an associated data management system to analyze student response data
2. A computer-assisted test construction (CATC) system containing 375 question-generating FORTRAN subroutines.
3. A library of 75 simulation and data reduction programs including 25 computer-simulated experiments.
4. A senior-level elective course, "Numerical Methods in Chemistry".
5. A library of computer-managed instruction programs.

The objective of this work has been to incorporate computing techniques where applicable in the chemistry curriculum in order to provide a partial solution to four problems facing educators. These are: remedial education including both inadaquate preparation and the inability of the student to maintain the pace of the general chemistry lecturer; stimulating and challenging highly motivated students; the composition of fair yet comprehensive examinations; and the data management problems associated with large general chemistry classes.

This paper contains a description of each of five applications, and a discussion of the specific ways computer-based techniques have enriched the chemistry curriculum at Pitt.

COMPUTER-ASSISTED INSTRUCTION

CAI is the application of computing technology to simulate the human tutor. CAI requires a responsive and reliable time-sharing system, an adequate number of terminals, a library of lessons, and the personnel required to maintain the software. Approximately 400 Pitt students interact each term with a DEC-System-1099 computer, using 50 public terminals that are located in four convenient sites around the campus. The system is available nearly 24 hours a day, seven days a week. The CAI library contains 45 lessons, written in the Pitt-developed languages, CATALYST and PIL. The author, with substantial assistance from both undergraduate and graduate students, wrote and maintains this library. Table 1 contains a description of five of the 45 lessons.

The subject content of the CAI library corresponds roughly to the coverage in a two-semester general chemistry course.

Table 1. Description of Five CAI Lessons

Lesson	Description
UNITS.1	First of three lessons that drill unit conversions, including °F ↔ °C ↔ K and metric ↔ English units of length and mass.
STOICH.3	Third of six lessons on stoichiometry; reviews percent composition and empirical formula calculations.
TITRAT.2	Second of three lessons on acid-base titrations; leads the student through the calculations involved in the titration of a weak acid with a strong base.
BONDING.1	First of two non-numerical lessons involving electron configurations, periodic trends, and chemical bonding.
NUKE.1	A lesson that drills energy changes in nuclear reactions.

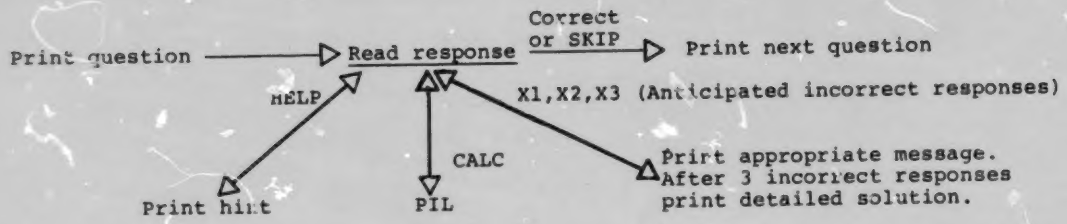

Print question
Read response
Correct or SKIP
Print next question
HELP
X1,X2,X3 (Anticipated incorrect responses)
CALC
Print hint
PIL
Print appropriate message. After 3 incorrect responses print detailed solution.

Figure 1. Branching Strategy for Each Question

Each of the 45 lessons contains between four and ten questions or problems. Each question or problem is generated using random numbers and randomly selected elements of arrays. For example, one of 15 nuclear reactions is randomly selected in NUKE.1. The branching strategy for each question is given in Figure 1.

A subset of the CAI library is used as a required homework exercise for students in the preparatory course. CAI is an optional resource for the other general chemistry students. On the first day of class each term, every general chemistry student is given a computer number and a 6-page handout that describes in detail the CAI lessons and the procedures for using them. Tables 2 and 3 provide an indication of student response to the CAI lessons. Chemistry 12 is the second semester of the two-term general chemistry sequence. Chemistry 10 is the preparatory course (one semester, no lab).

In both classes, more students indicated that the contribution of CAI to learning the course material was above average or exceptional than below average or very low. It is interesting that the texts (but not the GCEQ supplement) received the opposite evaluation in both classes.

In the past two years a CAI data management system has been developed and interfaced to approximately one-half of the lessons. This system writes a line of data to a disk file every time a student leaves a question in a lesson. The data include the student's computer number, the CAI lesson, the date, the

Table 2. Student Evaluation of Chemistry 12, Fall, 1975
To What Extent Did (Component) Contribute to Learning the Course Materials?

		Percentage Response					
Component	QPA[1]	Very Low	Below Average	Average	Above Average	Exceptional	Did Not Use
Lecture	2.8	2	2	28	52	17	-
Text[2]	1.8	9	30	37	22	2	-
Recitation	2.7	2	7	31	37	22	-
Laboratory	1.7	19	20	35	22	4	-
GCEQ[3]	3.4	4	2	6	31	57	-
CAI	2.5	-	9	11	22	6	52
Free Tutoring	2.1	-	8	23	8	0	62

[1] Very Low = 0, Below Average = 1, Average = 2, Above Average = 3, Exceptional = 4

[2] Dickerson, Gray and Haight, "Chemical Principles", Benjamin, 1974.

[3] Johnson and Epstein, "General Chemistry Examination Questions", Burgess, 1975.

[4] By teaching assistants in a special recitation room, approx. 20 hrs./wk.

Table 3. Student Evaluation of Chemistry 10, Fall, 1978
To What Extent Did (Component) Contribute to Learning the Course Material?

		Percentage Response					
Component	QPA	Very Low	Below Average	Average	Above Average	Exceptional	Did Not Use
Lecture	2.8	0.1	5	26	48	20	0
Text[1]	1.6	9	23	40	15	2	11
GCEQ[2]	3.1	0.6	5	11	44	39	1
CAI	2.6	6	8	24	41	19	1
Exam System[3]	3.3	2	0.6	13	34	49	1

[1] Newell, "Chemistry - An Introduction", Little, Brown, 1977.

[2] Johnson and Epstein, "General Chemistry Examination Questions", Fourth Edition, Burgess, 1978.

[3] Three hour-exams scheduled on Monday, during class. For each exam, an optional make-up exam was given the following Wednesday evening. The higher of the two scores was recorded for those who took both exams.

clock time (hour, minutes, seconds, and tenths of seconds), the question number in the lesson, and a set of integers that describe the student's responses in that question. Subsequent analysis of these data produce a table for each lesson with the percentage responses and timings for each question in the lesson. The following responses are recorded: correct answer, "help", "skip", three anticipated incorrect responses, all unanticipated incorrect responses, and if a detailed solution was presented. The lesson designer and his collaborators are now using these tables to revise and improve the lessons. If, for example, an unusually large percentage of students asked for help in a particular question, or failed to respond correctly in three attempts, then perhaps the question should be simplified. An analysis of the anticipated and unanticipated student responses will shed some light on common misconceptions and errors students make in learning chemistry. The significance of these statistical results will improve as large numbers of students use the materials over several years. This CAI library and data management system has the potential to be an instrument to measure and record the learning behavior of general chemistry students.

COMPUTER-ASSISTED TEST CONSTRUCTION

The multiple-choice exam is the most widely used tool to measure mastery of chemical concepts and skills in general chemistry courses. Two approaches to computerized generation of multiple-choice exam questions have evolved. One of these is to write several thousand complete items, prepare them in machine-readable form, and compile a bank of questions that can be selected using information retrieval techniques. The alternative, used here, is to write algorithms (FORTRAN subroutines) that generate exam questions. For example,

> What is the pH of a 0.827 molar acetic acid solution? (The pKa for acetic acid is 4.75).
>
> a) 3.83E-03 b) 2.42 c) 4.75
> d) 2.37 e) 11.58

The underlined parts of the question are variables. Another sample execution of this subroutine might read as follows.

> What is the [OH(-)] of a 0.150 molar hydrazine solution? (The pKb for hydrazine is 6.01).
>
> a) 0.150 b) 3.42 c) 3.83E-04
> d) 1.46E-07 e) 2.61E-11

The computer program that generates these questions contains instructions to calculate the correct answer and the four distractors. The Pitt CATC system currently consists of 375 question-generating subroutines. They are classified both by coverage and difficulty (easy, moderate, or hard). The current coverage is given in Table 4.

Table 4. Pitt's CATC System Coverage (Winter, 1979)

Topic	Easy	Moderate	Hard
1. Units and Conversion Factors	9	7	0
2. Atoms and Molecules	12	11	2
3. Chemical Reactions	3	5	7
4. Redox Reactions	4	4	0
5. Faraday's Laws	0	2	6
6. Colligative Properties	0	8	0
7. Heats of Reaction	4	5	3
8. Solution Stoichiometry	5	11	7
9. Mixed Stoichiometry	0	0	11
10. Laboratory Stoichiometry	0	4	0
11. General Gas Laws	6	10	1
12. Dalton's and Graham's Laws	2	4	5
13. Kinetic and Molecular Theory	0	0	6
14. Nomenclature	5	0	0
15. Solids	2	3	4
16. Atomic Structure	1	5	5
17. Bonding	0	19	5
18. Strong Acids and Bases	12	8	1
19. Weak Acids and Bases	4	20	8
20. Buffers	2	5	2
21. Solubility	2	6	10
22. Thermodynamics	0	6	14
23. Gas Phase Equilibria	2	3	2
24. Electrochemistry	0	5	8
25. Kinetics	4	1	6
26. Nuclear Chemistry	3	9	0
27. Coordination Chemistry	0	10	4
28. Organic Chemistry	4	0	0
29. Descriptive Chemistry and Misc.	4	0	1

The book, "General Chemistry Examination Questions"[1] (GCEQ), contains sample executions of this set of exam questions. The CATA system serves primarily as a time-saving device for instructors teaching general chemistry courses. It also has facilitated certain innovations, for example, repeatable testing.

Currently in the preparatory course at Pitt, three one-hour exams are scheduled for Monday, and are given during the class period. An optional make-up exam is scheduled for the following Wednesday evening. The exams are completely computer-generated. Although they are multiple-choice, a partial-credit scoring system has been devised. As the students leave the examination room they are given an answer matrix, containing the partial credits for each question, so they can compute their score. The class average is posted as soon as the machine scoring is completed. At least 75% of the class take the make-up exam, regardless of the average on the first exam. Most students improve their score on the make-up. The results of the most recent class evaluation were given in Table 3.

The GCEQ supplement and the exam system always receive the highest ratings on this evaluation. The use of GCEQ and the exam plus make-up exam system has a number of pedagogical advantages.

1. Exam anxiety is greatly relieved, at least on the in-class exam. The student can blow the first exam, study diligently, and probably improve his/her score on the make-up.
2. Students are aware in advance of the coverage, difficulty, and style of writing of exam questions. No study aid is more appreciated by a student preparing for an exam than a set of sample exam questions similar to those he/she will encounter.
3. Students have an opportunity to learn from their mistakes.
4. The make-up exam problem is solved. The instructor can have an arbitrary number of exams of equivalent difficulty and coverage generated by the CATC system.
5. The bad day syndrome is eliminated.
6. The GCEQ can be used to enrich the syllabus of a course. Much of the first semester of General Chemistry is review material for college freshman who have had one or two years of high school chemistry. The instructor can assign the

---

[1] K. J. Johnson and L. M. Epstein, "General Chemistry Examination Questions", Fourth Edition, Burgess Publishing Co., Minneapolis, Minn. 1978.

corresponding questions in GCLQ for review and self-study, and spend less lecture time on review.

7. For most students, the amount of time spent in concentrated study of the course material is directly proportional to the number of exams they take in that course. The make-up exam system is a positively reinforcing technique to get students to study harder. No student has ever complained about this additional work.

SIMULATION AND DATA REDUCTION PROGRAMS

The Pitt library of simulation and data reduction programs currently contains 75 FORTRAN programs. They are conveniently classified into three categories: function evaluation, data reduction, and computer-simulated experiments. The function evaluation programs allow students to enter the parameters of a mathematical model of a chemical system to discover the response of the system to arbitrary changes in those parameters. The data reduction programs extend the students calculating ability so that a more sophisticated treatment of experimental data is possible. Computer-simulated experiments are used to expose students to simulated data either as a pre-laboratory exercise, or as an enrichment experience for those experiments that cannot be done in the undergraduate laboratory because of time, safety, or expense constraints. Table 5 contains brief description of ten of these 75 programs.

Table 5. Ten Simulation and Data Reduction Programs

Program	Description
ACETIC	Simulates the determination of the weight percent of acetic acid in vinegar by NaOH titration. The student calculates the average value and the standard deviation.
EXCHG	Calculate the trajectories for the exchange reaction $H_A + H_B{-}H_C \longrightarrow H_A{-}H_B + H_C$ Parameters: initial energy, initial position, Morse potential constants.
HNATRN	Simulates the weak acid-strong base titration systems involving HA, H_2A, and H_3A. Parameters: number of acidic protons, concentration and volume of acid, acid dissociation constants, and concentration of base.
KA	Simulates the spectrophotometric determination of the dissociation constant of an indicator.
KC	Simulates the equilibrium system $A + B \rightleftharpoons C$ The student determines the equilibrium constant from spectrophotometric data.
KINETI	Calculated concentration-time profiles for the reaction $A + B + C \longrightarrow$ Products The student determines the rate law and the activation energy of the reaction.
QUAL	Simulates a subset of the inorganic qualitative analysis scheme. The student identifies 4 of 13 cations from groups 1, 2, and 3.
NEUTRO	Simulates the analysis of an unknown binary mixture by neutron activation analysis.
NMR	Simulates AB, AB_2, ABX, A_2X_2, and A_2B_3 NMR spectra. Parameters: chemical shifts and coupling constants.
TITR	Calculates the two thermodynamic dissociation constants of an amino acid. TITR reads student pH-volume of titrant data and uses the Debye-Hückel law to calculate the activity coefficients.

The most extensively used programs in this library are QUAL and TITR. The students in the second term of general chemistry study a 13-cation qualitative analysis scheme in the laboratory and find QUAL particularly helpful for practicing the separation and identification procedures before they are graded on them in the laboratory. TITR provides a considerably more sophisticated analysis of titration data in a quantitative analysis course than could be feasible without the computer. The author has used computer-simulated experiments (for example, ACETIC, KA, KC, KINETI, and NEUTRO in Table 5) for required and optional homework assignments in his general chemistry courses. These assignments are individualized and self-grading. The student logs on to the time-sharing system, runs the program, enters his/her name which initializes a random number generator, and receives the data for that experiment. He/She then logs off, performs the analysis off-line, and returns to run a corresponding answer program which evaluates his/her results. If the results are correct within an arbitrary tolerance, the student hands in the answer program's congratulatory message and his calculations. Otherwise he/she continues working on the analysis of his/her data. Some of these programs are fairly sophisticated and prove particularly useful for challenging highly motivated students.

"NUMERICAL METHODS IN CHEMISTRY"

This is an elective course, designed to bring numerical methods and programming techniques to bear as tools to solve a set of chemistry problems not usually encountered by undergraduates. The prerequisites include an introductory FORTRAN programming course and calculus. The enrollment consists primarily of seniors, some juniors, and a few graduate students. The content of the course is indicated in Table 6.

There are weekly homework assignments and two major projects, one due at the mid-term, the other at the end of the term. These projects involve writing and documenting a computer program to solve some problem of interest to the student. Some of these student projects are listed below.

1. Calculate the titration curve for H_3A in the presence of CO_2.
2. Determine molecular parameters from published microwave spectroscopy data.
3. Use NMR spectrum generating programs to assign an unknown spectrum.
4. Simulate complex kinetic schemes, for example, combustion and photochemical smog systems.
5. Solve complex aqueous and gas phase equilibrium systems.
6. Calculate the trajectories for a hydrogen atom - hydrogen molecule exchange reaction by solving Hamilton's equations of motion.

Most of the students document these programs beautifully. The sample executions are usually accompanied by computer-generated plots and a detailed discussion of the results.

No cake elective, this course attracts only highly motivated students. They realize that this course provides a unique opportunity to apply powerful computing and mathematical techniques to some interesting chemical systems.

COMPUTER-MANAGED INSTRUCTION

The major effort in CMI is the data management system for the CAI lessons mentioned previously. Two programs, PROMPT and SCORER, are used in conjunction with exam generation and scoring. PROMPT is an interactive program that prompts the instructor for the partial-credit matrix which is subsequently used by SCORER to compute the scores on multiple-choice exams. SCORER prints an item analysis and lists of student scores, sorted by identification number and score. Other programs have been written to choose and record the higher of two scores if an exam and make-up exam have been given, and to maintain a master file of student scores. The program GRADER reads this master file at the end of the course, applies weighting factors to the component scores, computes the total weighted score, and prints sorted lists of student scores and a histogram.

Such programs not only save instructor time, but also can contribute to the effectiveness of teaching. The CAI data management system provides quantitative information on student interaction with chemistry concepts and problems. Partial-credit scoring of multiple-choice exams is only fair since partial-credit would certainly be given if the exams were scored by hand. An item analysis is a very useful feedback mechanism not only to identify poor exam questions, but also to identify concepts that were not adaquately covered in lecture or in the text.

DISCUSSION

How can the computer be used to help solve the problems of remedial instruction in chemistry? A good CAI library can help students acquire mathematical skills and learn chemical concepts. Many inadaquately prepared students who are reluctant to demonstrate their ignorance to an instructor are willing to interact

Table 6. Contents of "Numerical Methods in Chemistry"

1. The Computer and FORTRAN Programming Techniques
2. Programs with Closed-Form Algorithms
3. Roots of Equations
4. Systems of Linear Simultaneous Equations
5. Regression Analysis
6. Numerical Integration
7. Differential Equations
8. Eigenvalues and Eigenvectors
9. Miscellaneous Topics

with a time-sharing system. The computer is available, resourceful, patient, and reinforcing.

How can the computer be used to challenge and stimulate highly motivated students? A set of simulation and data reduction programs can extend the curriculum in any undergraduate chemistry course. The elective course, "Numerical Methods in Chemistry", provides students with powerful tools to probe chemistry concepts that might not otherwise be covered in the undergraduate curriculum.

How can the computer be used to help instructors compose good examination questions? A CATC system can be iteratively improved until a reliable set of examination questions specified by topic and difficulty can be obtained. A CATC system can also support educational innovations requiring repeatable testing.

How can the computer be used to free instructors from clerical tasks, and reduce the burden of tutoring rather than stimulating students? A CATC system can reduce by a significant factor the time required to write exams. A CAI library can assist him/her in the remedial tutorial-drill interaction so desparately needed by inadequately prepared students. A CMI library can significantly reduce the clerical component of his/her job.

The computer, properly used as multi-purpose educational tool, can enrich the chemistry curriculum.

The Dartmouth CAUSE Report

Chaired by Bill Arms
Dartmouth College

The CAUSE Project Seen from the Computation Center

Bill Arms

A key factor in the success of the Dartmouth CAUSE project has been that advanced computing has been carried out by people whose level of previous experience was not particularly great. Writing interrupt-driven programs, hardware/software interfaces, and telecommunications protocols are not normally handled by faculty members primarily interested in the computer as a tool for teaching.

The project was developed by faculty members for their educational purposes, and this seems the natural way in an academic environment. But for the early commitment of time by a member of the Office of Academic Computing (Arthur Luehrmann), the coordination necessary to get the project going would have been lacking.

The Computation Center was not involved in the early development of the project, but as the project has matured, a need has arisen for routine services that are much better handled by an administrative organization, such as Computing Services, rather than by academicians. Typical examples are the publication of manuals, storage and maintenance of both software and hardware, and turning some routines from experimental to long-term production versions. Now that the NSF CAUSE money is coming to an end, the momentum could easily be lost if the individual academic departments have to rely on their own resources; but coordination between Computing Services and the department could make this into a long-term productive venture.

There is one final comment that has been made. All the leaders of the CAUSE project have been tenured professors. Cynical people have suggested that this is because untenured faculty are reluctant to tackle educational projects, but I am prepared to be more optimistic and suggest that the sort of work that has been done requires considerable maturity to bring subject specializations, educational ideas, and computing together.

Use of the Minicomputer in the Undergraduate Biology Labs

Thomas Roos

We have used on-line computers to collect and analyze data from biochemical and physiological experiments in which speed, choice, and capacity determine the success of a procedure. Students in physiology and endocrinology have used the computer in series with a signal generator, an oscilloscope, and a terminal in order to obtain data, check them for suitablilty, preprocess them to remove excess points before and after the desired recording periods, store the data, and analyze them.

Students have measured the kinds and quantities of steriod hormones present in blood and tissues following experimental manipulation by extracting the hormones, separating them by chromatography, and detecting the amount and location of the separated spots using reflectance, multi-frequency spectrophotometry. The computer accumulates data for blanks, standards, and samples at three frequencies, separating complex curves into single, analyzable peaks and compares the peaks of samples and standards. It reports the likely matches between any sample and the standards using migration and absorption data as criteria.

Another use of the computer allows students to collect and display a nerve trace or nerve-muscle trace on an ordinary oscilloscope. Data enter the computer through the oscilloscope, used as a preamplifier, and are then plotted to the oscilliscope screen. If unsatisfactory, the curve is repeated. Stored data can be analyzed and accumulated. Simultaneous recording from adjacent nerves or down-stream effectors (muscles and glands) can be stored blindly and then called for subsequent display or temporal analysis against the nerve trace. The effects of drugs, stimulus frequency, temperature, and other experimental influences can be stored on the same diskette and subsequently compared directly with the basal trace.

Use of the Minicomputer in the Introductory Physics Course

Elisha Huggins

One of our main goals in the laboratory part of the Introductory Physics course is to teach effective techniques for handling experimental data. Today that means to teach students to use the computer to handle data files and do such operations as least square fitting and numerical integration.

Our first task in carrying out this goal was to select an experiment that is interesting and understandable in its own right, but that clearly demonstrates the capabilities of computer analysis of data. We selected Fourier analysis of notes from a musical instrument because: 1) Fourier analysis of realistic data is not possible without a computer, 2) the harmonic construction and analysis of musical notes can be demonstrated both on a visual and on an audio basis, 3) the concepts of Fourier analysis apply throughout the physics curriculum, and 4) the experiment is self-checking because the student can reconstruct the original curve from the calculated curve.

To carry out the Fourier analysis experiment, we had to develop a way of teaching the topic at the freshman level, and to efficiently get experimental curves into the form of a data file in the student's catalogue. How easy it is is to teach Fourier analysis will be demonstrated in the discussion.

Oscilliscope-Controlled Minicomputer

To make the minicomputer easily accessible and function in a way that is natural in a laboratory environment, we constructed a system in which the minicomputer is controlled by a standard laboratory oscilloscope. When the student presses the "data acquisition" button on the scope, the computer reads the next sweep and displays the acquired curve on the monitor scope. If the "save" button is pressed, the curve is sent by phone to the main campus computer and stored in the student's catalogue for later analysis. During data acquisition, the minicomputer samples the scope's deflection plate voltage at a rate of 200 points per sweep, so that the student stores exactly the curve he sees on the scope and can take full advantage of the scope's amplifier and trigger circuitry. Once a student has learned to use an oscilliscope, he can immediately use the data acquisition system.

The Minicomputer in Undergraduate Chemistry Laboratories

James F. Hornig

Although mini- and microprocessors are becoming increasingly evident in chemistry laboratory equipment, chemistry students seldom have the technical background to appreciate the role of the computer in a particular instrument. It is the goal of the Chemistry segment of the Dartmouth CAUSE project to develop laboratory applications that are sufficiently familiar and simple that students can grasp readily the operation of the computer. At the same time we have sought applications in which the power of the computer is evident.

One of the most traditional laboratory operations, the acid-base titration, has turned out to be an extremely successful application. A variety of possible laboratory problems quickly introduce the student to the possible roles of the computer in a lab.

In the entire sequence of laboratory activities the computer never becomes just a black box. The physical layout of the instrument is such that students get a clear picture of the interconnections of the various components. Although no direct programming is involved, the use of interchangeable diskettes for storing programs and data, as well as the necessity that some programs be compiled before use, give students a considerable conceptual familiarity with the operation of the instrument.

Finally, the bridge from simple to complex applications is completed when students become aware that the same apparatus and the same algorithms are being used in an active research project.

Classroom Information System

G. Christian Jernstedt

The classroom information system provides students and instructors in large (or small) courses with a powerful means of acquiring, analyzing, and displaying information in real time. Using a microcomputer, custom data acquisition and processing modules, floppy disks, and television displays, the system operates in a conversational mode to provide the instructor with a wide range of input and output options.

The system input is composed of individual response boxes which accept information from each individual student or from other organisms such as experimental animals. Feedback can be returned to the individual, and the collected group or individual data may be analyzed and displayed on television monitors in the classroom.

With such a system, instructors can poll students, conduct experiments, and operate demonstrations, all while allowing the students to suggest data analysis and display modes. The result is a highly motivating interaction among students, their instructor, and the data or organism being studied in the course.

A DESCRIPTIVE APPROACH TO THE TEACHING OF COMPUTER ORGANIZATION

Zvonko G. Vranesic
Departments of Electrical Engineering and Computer Science
University of Toronto

Abstract

This paper addresses three issues involved in teaching of courses on computer organization. The first deals with the desired degree of emphasis placed on engineering aspects of the subject matter. The second involves the choice of material that should be covered. Finally, the relative merits of a descriptive approach to teaching such courses are considered.

1. Introduction

A fundamental course in a computer science curriculum is a course covering the structural principles of computer hardware. There is a lack of consensus about the choice of the material that should be taught, the relative emphasis between programming and electronic aspects that ought to be stressed, and the type of presentation that is best suited for teaching the subject matter.

This paper discusses one point of view about such courses. The opinions expressed are largely influenced by the experience of teaching students in computer science and electrical engineering at the University of Toronto, and by the feedback coming from the industrial community which ultimately absorbs the graduates of the University.

2. Computer Architecture vs. Computer Organization

Computer architecture deals with the functional structure of hardware as seen through the eyes of a programmer [1]. This includes characteristics such as wordlength, bus structure, number and type of registers, addressing modes, instruction set, input-output mechanism, and memory access techniques. Thus, one can speak of the IBM-370[1] architecture, the PDP-11[2] architecture, etc. In contrast to the architecture, a specific implementation of a given machine, for example a particular model in the PDP-11 line, is computer organization. It involves some hardware details that are essentially transparent to the programmer. These must include the control mechanism and associated signalling, bus interfaces, internal structure of storage units, etc. Moreover, it should involve interfacing details of the related peripheral devices.

It has been a fairly common practice to place greater emphasis on computer architecture. This is an unfortunate development that is the result of insufficient importance being placed on teaching computer hardware. Since few people in the computer business came into direct contact with hardware, it was easy to argue that computer specialists would benefit the most from hardware exposure at the architecture level, concentrating their efforts on software aspects. Adequate treatment of computer organization tended to be restricted to engineering-oriented programs, which on the other hand often underplayed the importance of software.

Then, microprocessors came along. Quickly it become apparent that the most useful people for microprocessor system development were neither software nor hardware specialists, but those who have a reasonable knowledge of both areas. When one looks at microprocessors at the chip level, it is just as important to know interfacing details as it is to understand the architecture. Thus, a shift towards computer organization is indicated in the microprocessor environment. It should also be noted that it is hardly appropriate to talk about significant microprocessor architectures that reflect themselves in a number of distinct implementation models. For example Intel 8080 and 8085 are supposed to have the same architecture. This is so if the 8085 is used with directly compatible components such as the 8155 and 8355. It is questionable whether the architecture is the same when other general purpose memory and peripheral chips are used. A more powerful microprocessor in the Intel family, the 8086, while capable of executing the 8080 instruction set, has a significantly different architecture.

Microprocessors are now used in laboratory work associated with courses on computer architecture. Here they reinforce the need for including adequate electronic details in the course material. Thus, it is reasonable to argue that courses with a strong emphasis on computer organization, incorporating practical hardware details, should be given high priority in computer science curricula.

3. What Should Be Taught

The choice of topics that should be taught is of great importance. Much has been said and written about this question [2]. While it is easy enough to say that the most relevant material must be covered, there is considerable disagreement as to what this material is. In fact, it is readily apparent from looking at available text books that aside from a few basic topics, most people seem to have their own ideas as to what is important and interesting.

There is a tendency to spend much time on well-defined concepts, particularly those where some formalism can be introduced. A good example is computer arithmetic. It lends itself to neat presentation, where circuitry needed to implement the desired algorithms can be described in terms of concise and easily understood expressions. Moreover, some of the useful algorithms are rather ingeneous and fun to talk about, and suitable illustrative examples are readily available. Another popular topic is the internal structure of memories. There is general agreement on the relative merits of the few schemes for constructing primary memories and they provide nice material for classroom discussion.

---

[1] International Business Machines Corp.

[2] Digital Equipment Corp.

On the other hand it is much more difficult to adequately treat input/output techniques, interfacing problems, communications and control signalling, with the associated timing and signal level constraints. Yet, these are the main areas of concern in digital design, particularly in small machine environments. Few graduates are likely to spend much time on arithmetic units, either designing them or having to understand their peculiarities. Most of their hardware efforts (since the large majority will be involved in the development of software) will be focused on interfacing and signalling problems. Where no clear-cut best solutions exist, one should try to produce a sufficient number of examples to illustrate the alternatives. Design practitioners are the best source of the required information.

Implementation (and use) of digital hardware can be a frustrating process. The student should learn something about the difficulties in implementation. This is best done in a suitable laboratory. It is useful to consider the problem of testing, which is often naturally covered in the course of laboratory experiments. This task is more difficult in courses that have no laboratory work associated with them. But even then it is worthwhile to convey that the world of real machines is full of problems that may not be apparent from the pages of a textbook. A great many difficulties are caused by the imperfect behaviour of physical devices, making it wise not to overemphasize the value of simple models.

Computer hardware, and computers in general, are often treated in a far too complicated fashion. There is nothing that is inherently difficult about computers conceptually. There are a few very useful concepts that should be taught in as simple terms as possible. It is advisable to avoid the tendency to complicate relatively simple concepts, no matter how powerful or useful they may be. Microprogrammed control is a good example of this.

4. Descriptive Approach

There are several different approaches to teaching computer organization. Instructors tend to be strongly opinionated about the best approach, noting that their opinions are often influenced by the flavour of the textbooks that are available and seem appropriate for the course in question. The following discussion is an attempt (by a somewhat biased author) to consider the relative merits of one possible approach, namely that of relying upon descriptive examples that are closely related to real, commercially available machines. A descriptive approach has a number of advantages, but also some drawbacks.

4.1 Advantages

It is impossible to do justice to a course on computer organization without the aid of good examples. The examples should contain sufficient details to ensure that the students will appreciate both functional and implementation intricacies of the aspects that are being discussed. It is very easy to fall into a trap of never getting past the level of grossly oversimplified block diagrams and generalities. The students are prone to assuming that structures represented by block diagrams are intuitively obvious, and general comments often mean little to them. This undesirable situation is best avoided by using meaningful examples from the real world [3].

A major advantage of using examples derived from specific computer architectures that are produced by manufacturers (e.g. PDP-11, IBM 370, HP 3000[3], etc.) is that students become familiar with at least some commercially available products. This is clearly desirable now when practical aspects of university education are being emphasized.

Computer design, and digital system design in general, is still not a science in the traditional sense, it is an art. This means that optimal solutions rarely exist. The design process is not unlike most engineering design endeavors. Many design alternatives are usually possible, each presenting some positive and some negative factors. How one goes about making sensible decisions in this process is ideally what students should be taught. This can be done well only by looking at real examples that incorporate typical constraints. A study of real designs leads to a better appreciation of trade-offs involved. In addition to understanding the architectural structures, it is possible to gain some feeling for electronic and economical aspects as well. Functionality, speed of operation (or performance), maintainability, and cost are basic criteria that can be properly discussed within the framework of a real machine.

If laboratory work is associated with the course (as it ideally should be), then it makes sense to use examples based on the computers used in the laboratory. Thus, students not only become acquainted with the structural concepts but also gain some hands-on experience, which typically leads to insights into related side issues.

The alternative to using real machines as examples is to define a hypothetical machine. It is quite easy and tempting to come up with a paper design of a computer that can serve as an illustration for most concepts taught in the course. Indeed many textbooks follow this approach, defining not only complete hypothetical computers but often also fairly extensive assembly languages for them. The trouble is that such machines usually abound with features that do not exist in that combination in any commercial computer. The features included are often incompatible and difficult to implement. Sometimes one runs across features whose implementation is not economically feasible. Occasionally, outright design errors are encountered.

---

[3] Hewlett-Packard Co.

Another problem is that it is impossible to get any hands-on experience with hypothetical machines. While it is possible to simulate them, this is hardly an adequate alternative.

4.2 Disadvantages

The most serious difficulty with the descriptive approach is that no single computer exists that illustrates all of the desired concepts. Moreover, just about any computer that one chooses is likely to have some unappealing features. In addition, if a single machine is being used, it can easily happen that the students leave with the impression that no other viable machines exist. If asked (as an interesting exercise) to suggest a design for a simple computer of their own, it is not unlikely that their proposals will closely resemble the example discussed in the class.

The above drawback can be avoided by drawing upon examples from more than one computer architecture. For instance one might use the PDP-11 architecture as the basic framework of discussion, illustrate the input/output processor schemes through the IBM 370 channels or the peripheral processing units in the Control Data machines, support the presentation of stack computers with the HP 3000, etc. If the required material is carefully extracted and not clouded with unnecessary details, the students will benefit from such exposure to machines that they are going to run into upon graduation.

Discussion of commercially available architectures can be carried to a dangerous extreme, whereby general principles are totally masked by the details of particular examples. This typically occurs in courses that are based exclusively on microprocessors, especially if the same microprocessors are also used in laboratory work. One should not forget that microprocessors and microcomputers are really just ordinary computers that happen to be manufactured on one or more LSI chips. Thus, it is much more useful to talk about general computer principles, while relying upon the laboratory to place the characteristics of the microprocessor environment into proper perspective.

Finally, it is necessary to consider some of the objections that are raised by the academic purists. One often hears that a descriptive approach, or a case study approach as it is sometimes inappropriately refered to, is not scientific in nature. Indeed, some highly prominent computer scientists profess that computer hardware design is immature as a scientific discipline. They claim that this immaturity is characterized not only by recurring errors in new designs (caused by insufficient understanding of fundamental principles and past mistakes), but also by preferring pictorial representations of hardware to symbolic and word descriptions. This author finds it difficult to get excited by such arguments. At times they seem to be quite frivolous. Computer design is essentially an engineering endeavour. As such it is subject to great many practical constraints, with scientifically pure goals often being economically unfeasible dreams. It is probably the best to accept this reality, at least for the time being, and try to ensure that we succeed in teaching our students what present computers are really like, since ideal machines are still a long way away.

5. An Illustrative Example

It was suggested in the preceding section that the descriptive approach to teaching computer organization should not be synonymous with doing a case study. Of course, when a real machine is used in examples its features should not be distorted in any way. This is particularly important where characteristics that have a direct impact on software are concerned. For example, it is not advisable to tinker with the instruction set, since it is heartening for students to know that whatever programs they are asked to look at or write can be executed on the machine that they are learning about. On the other hand, it makes much less sense to cover all implementation details of specific hardware that can be rather cumbersome. Suitably simplified or generalized schemes derived from the chosen architecture lead to a more purposeful presentation. In other words it is advisable to deviate from the exact case study.

Consider the diagram in Figure 1. It depicts an important facet of microprogrammed control: the way in which the

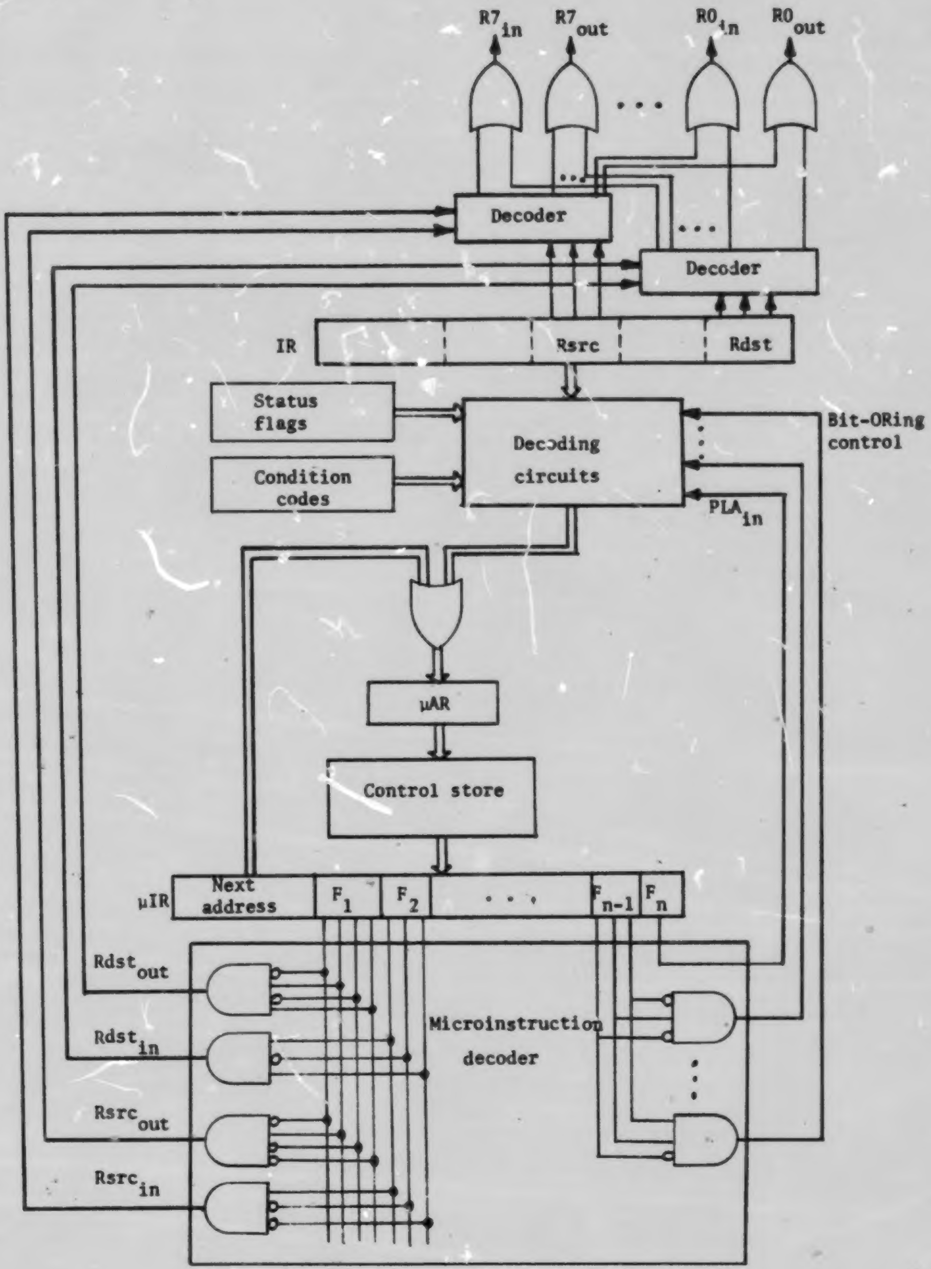

R7in
R7out
R0in
R0out
Decoder
Decoder
IR
Rsrc
Rdst
Status flags
Condition codes
Decoding circuits
Bit-ORing control
PLAin
μAR
Control store
μIR
Next address
F1
F2
Fn-1
Fn
Rdstout
Rdstin
Rsrcout
Rsrcin
Microinstruction decoder

Figure 1. Generation of control signals

required control signals can be generated. The scheme is patterned after the PDP-11 structure, but it is sufficiently simplified (and different from actual PDP-11 implementations) to serve as a general example. It shows how signals that control the gating of registers are obtained from the information in a microinstruction and a machine instruction held in the instruction register. It also indicates a widely used method of microprogram sequencing control. A field in the microinstruction is used to specify the address of the next microinstruction, which may be modified through bit-ORing circuitry.

This example is intended to show the flavour and the level of presentation that may be appropriate. Actually, it contains sufficient information to allow a fairly intensive and general discussion of some of the key aspects of microprogrammed control. Such examples can be developed to demonstrate most of the relevant concepts in computer organization. The effort involved may be considerable, as discovered by this auther during the process of coauthoring a recently published book [4], but it is worthwhile if the positive reaction of the students who have been exposed to such material is a useful indicator.

6. Concluding Remarks

The descriptive approach is by far the most practical approach to teaching of computer architecture and organization. The structure of computers can be handled as a collection of concepts and organizational schemes that can be suitably illustrated with examples derived from widely accepted commercial products.

An excessively formal introduction is not desirable at this stage of development. Hardware description languages are suspect in their educational effectiveness. Students easily find themselves more concerned about the syntax of the language than about the structures that they are used to represent. Languages are more effectively used in documentation and simulation tasks.

Perhaps the most important question is whether or not it is possible to break down general computer structures into a set of basic primitives. Few results have emerged so far to indicate an affirmative answer. The desired level of primitives ought to be considerably above the level of primitives used in basic logic design and switching theory. When attempting to define this level we should keep in mind the fate of switching theory which in just a few years went from a great activity and relevance to a subject that is of little practical interest. Formal structures may find it difficult to keep up with the developments in a rapidly changing technological area.

7. References

1. C.G. Bell and A. Newell, "Computer structures: readings and examples", McGraw-Hill Book Co., 1971, p. 562.

2. G.E. Rossman et al, "A course of study in computer hardware architecture", Computer, vol. 8, pp. 44-63, Dec. 1975.

3. S.G. Zaky, Z.G. Vranesic and V.C. Hamacher, "On the teaching of computer organization to engineering undergraduates", IEEE Trans. on Education, vol. E-20, no. 1, pp. 27-30, Feb. 1977.

4. V.C. Hamacher, Z.G. Vranesic and S.G. Zaky, "Computer organization", McGraw-Hill Book Co., 1978.

COMPUTER LITERACY FOR LIBERAL ART STUDENTS:

THE CASE OF COMPUTER MAPPING

By A. Rees Clark, Ph.D.
Colgate University, Hamilton, New York 13346
(315) 824-1000, 549

Computers allow college teachers to enhance curricula with concepts and procedures that could not be effectively addressed before computerization. Computer enhancement is a more exciting development than the more trivial reassignment of rutinary instructional and administrative chores. The computer literacy achieved in computer-enhanced programs may promote the goals of liberal education while expanding the technical capabilities of students in their major fields. This paper will describe a simple conceptual model of computer literacy and show how the model has been applied in geography.

Computers have assumed a role in American society that has made their needs competitive with human needs in a wide array of production, distribution, and decision-making contexts. A minimal familiarity with computers is at least as central to the life of a modern American as, say the three-fifths compromise, Manifest Destiny, or the Triple Entente. Ten years experience at four different universities with four different brands or configurations of computer systems suggests to me that relatively few students in general liberal arts programs achieve significant understanding of the actual and potential roles of computers in society. At the risk of seeming facetious, I believe this is because the computerization of American universities has been managed by people who are comfortable with and enjoy using computers. Such computer professionals or researchers are rarely solicitous and often intolerant or contemptuous of persons who do not share their enthusiasm of the enhanced role of the computer. If this attitude manifests itself in poorly documented, esoteric, and inaccessible software, the liberal arts student's first contact with the computer is usually his/her last voluntary visit. For many students the computer becomes not simply a black box, but a vaguely threatening enigma. Such students do not learn to control computers and may even join the large segment of the population who feel they are ruled by them.

Computer Literacy

Why should an average person know anything about how to use computers or how they work? Simply put, computer literacy allows one to challenge the system. Many of the familiar social conflicts over computerized tasks--automated billing, bank statements, etc.--may reflect the juxtaposition of an incompetent programmer and an ignorant client, rather than some limitation on the capabilities of computers. The computer literate knows the differences between these sources of error and argument. This is a critical faculty on a par with effective reading, communication, and analysis in a world in which the computer is the intermediary for so many administrative contacts.

Computer literacy is developmental. Levels of literacy are analogous to levels of development of speech or writing. The toddler is interested in performing rather straightforward tasks and cares little for analysis of procedures. The sophisticated adult may want to ponder relationships between classes of data structures, tasks, procedures, and languages. A model for cultivating computer literacy in college students should reflect what is known about developmental and experiential learning and probably has little to do with the interests of formal computer scientists. At least at the beginning, the computer should be introduced into the curriculum as a normal part of the ongoing activities of many disciplines under the assumptions that (1) average people doing a variety of tasks use computers every day, and (2) one doesn't have to be particularly good at mathematics to use computers productively. Conventional Introduction to Computer Science texts start too close to the machine to qualify as texts on computer literacy for students whose interests in computers are either very general or lie exclusively in disciplinary applications. Our hypothetical average student will bail out at the very 0000000000000001st binary number. Instead of understanding how computers work, computer literacy is simply the ability to exchange information with computers at a level appropriate to the problem the user wants to solve. At every level, some aspects of the computer will remain a black box. In the rest of this paper I will describe a five-level model of development of computer literacy and show how it has been applied to computer cartography.

Levels of Computer Literacy

(1) The toddler level. At this level the user learns two things. One is how to establish and terminate communication with the computer. The other is how to initiate execution of some program that will perform a desired task, requiring no decisions about program function. This level presupposes no knowledge of machine concepts and no knowledge of language constraints (the classic example: FORTRAN data format conventions). The machine simply accepts data (numbers or natural

language words) and produces the result.

(2) The child level. At this level the user can use programs that require decisions about program function but still require no decisions about computer function. He/she can manipulate the program using an appropriate command structure. Well-known examples of programs accessible at this level include SPSS, BIOMED, and SYMAP.

Levels 1 and 2 both allow the accomplishment of tasks without knowing how computers work. Variety and complexity are introduced by the nature of the problems. Each level could, of course, be further subdivided.

(3) The adolescent level. Now the user has hit an impasse. Some task--or more likely a variant of a simple task--cannot be accomplished by a canned program. The user wants to modify a program function by changing the source file. This can be done in two ways. The user can turn to a professional programmer and pay for the service, or he/she can learn elementary programming. A common entry point to this level is the need to manipulate data sets to render them acceptable to programs that can be run at the child level. The student who chooses not to learn a programming language has acquired considerable computer literacy but will remain incapable of independent computerization of problems.

(4) The adult level. Now the user can write complex source programs in a standard high-level language to solve applied problems within the constraints of state-of-the-art software systems. Because the adult is a programmer, he/she is in a position to know how procedures are created and whether broadly comparable tasks are computable. Such a student is no longer vulnerable to capricious decisions by a programmer who might otherwise provide an unsatisfactory algorithmic solution to a problem and claim that that was the best that could be done.

(5) The virtuoso level. Relatively few students will go on to study the properties of computers themselves. The highest level of computer literacy allows one to design systems to solve classes of problems. The design of a library of statistical programs or a set of mapping and spatial analysis programs with broad compatibility of data structures and command languages can be accomplished at this level.

There is really a sixth level of development, but it goes beyond literacy and is reserved generally for the computer professional. It involves the study of machine languages and machine architecture and the comparison of hardware and software systems. Most computer-enhanced education can be accomplished at the first two levels. Most research is done at levels three and four. The model might be compared to a nested set of Chinese boxes; each box may be examined in detail and considerable pleasure taken from it without opening the next.

A Simple System for Computer Literacy in Cartography

The model is being used to develop computer mapping software at Colgate University. Colgate is a liberal arts college with some 2,500 students. About 400 students per year take geography courses, of whom about half deal with some computable spatial problem. We have evolved a small spatial analysis program library. Our computer is a DEC-System KA10 with about 40 scattered terminals including a drum plotter and a Tektronix 4014-1 storage tube graphic display terminal.

Program development has been guided by the principle that solving a small problem is better than talking about a large problem. Accordingly, our library provides for simplified analyses of common geographical problems using fairly standard techniques. Data structures are kept compatible by faculty fiat; student- and faculty-contributed programs are written under top-down programming guidelines. For example, our regression program produces a file of residuals that can be read by a group of mapping programs. Inevitably, some parts of the system work better than others.

One goal of the geography program library is to make computer mapping available to the whole college community, and the literacy model has been applied to enable this. Students in introductory geography (and certain other disciplinary) courses can make maps at the toddler level by simply showing up at the computer center with appropriate data and a one-page write-up on the toddler-level use of a program called CONMAP (for CONversational MAPping Program). CONMAP is a scan-line technique producing choropleth maps on the lineprinter (after Scripter, 1966).

At the toddler level (this terminology is replaced by a euphemism more conducive to ego building) the student merely types .RUN GEO:CONMAP. The program asks the student if he/she would like the computer to do as much of the work as possible. If the user says yes, the only imput required by the program is (1) the name of the file containing the base map (available from a printed list), and (2) the series of values (e.g., the population density of each of the 62 counties of New York) in alphabetical order of the area names.

Next the student is given the chance to make as many maps as he/she wants on this or other available base maps. (Frequently used base maps like the United States or states in the Northeast are kept on-line; others are available on tape on short notice and can be placed on line by a faculty member or student cartography assistant.) If the student indicates that no more maps are to be made, the program displays instructions for directing output to the lineprinter. The student logs off and collects his/her map(s) at the computer center where they are waiting, well-labeled, in a delivery area (Figure 1).

Under the time-sharing framework, a simple mapping problem can be completed very quickly, making the program both popular and easy to include in course planning. The connect time necessary to prepare the map in Figure 1 was approximately five minutes. An unsophisticated student normally requires about 15 minutes connect time for his/her first map; most of this time is spent in data entry. CONMAP has been in use at Colgate for two years in geography courses. Introductory computer science courses have also used the input requirements as a framework for teaching manipulation of data matrices.

More sophisticated students can waive the do-it-all option and control any major function they wish. Functions they cannot or do not want to control retain the default condition. Students can thus progressively control symbolism, classification, data entry, and base map construction, together or independently (Figure 2).

Students at the adolescent level have gradually contributed embellishments to the program. Students in cartography courses contribute regularly to the growing library of base maps (mostly scaled and designed with term paper applications in mind).

Now under development is a companion program that uses the same data structures to produce perspective profile diagrams (3-dimensional maps) such that a statistical surface can be viewed from any superior position (Figure 3). The same concept of levels of control is applied with relatively simple data requirements to produce the map either on the terminal or on the drum plotter.

Naturally, to gain the speed, accessibility, and facility of these programs, we have consciously sacrificed some scope and flexibility. Much more splendid mapping programs exist, and we introduce them later in our cartography curriculum. Our attention has been focused on the beginner who comes to scientific analysis in human geography with a certain trepidation about quantitative analysis in general and computers specifically. Giving the student immediate apparent control over the computer, we have found, increases his/her zeal for rigorous approaches to complex problems and promotes confidence in his/her own powers.

REFERENCES

Scripter, M.S. (1966) "Choropleth Mapping on Small Computers," Proceedings, Assn. of American Geographers, Vol. 1, pp. 133-36.

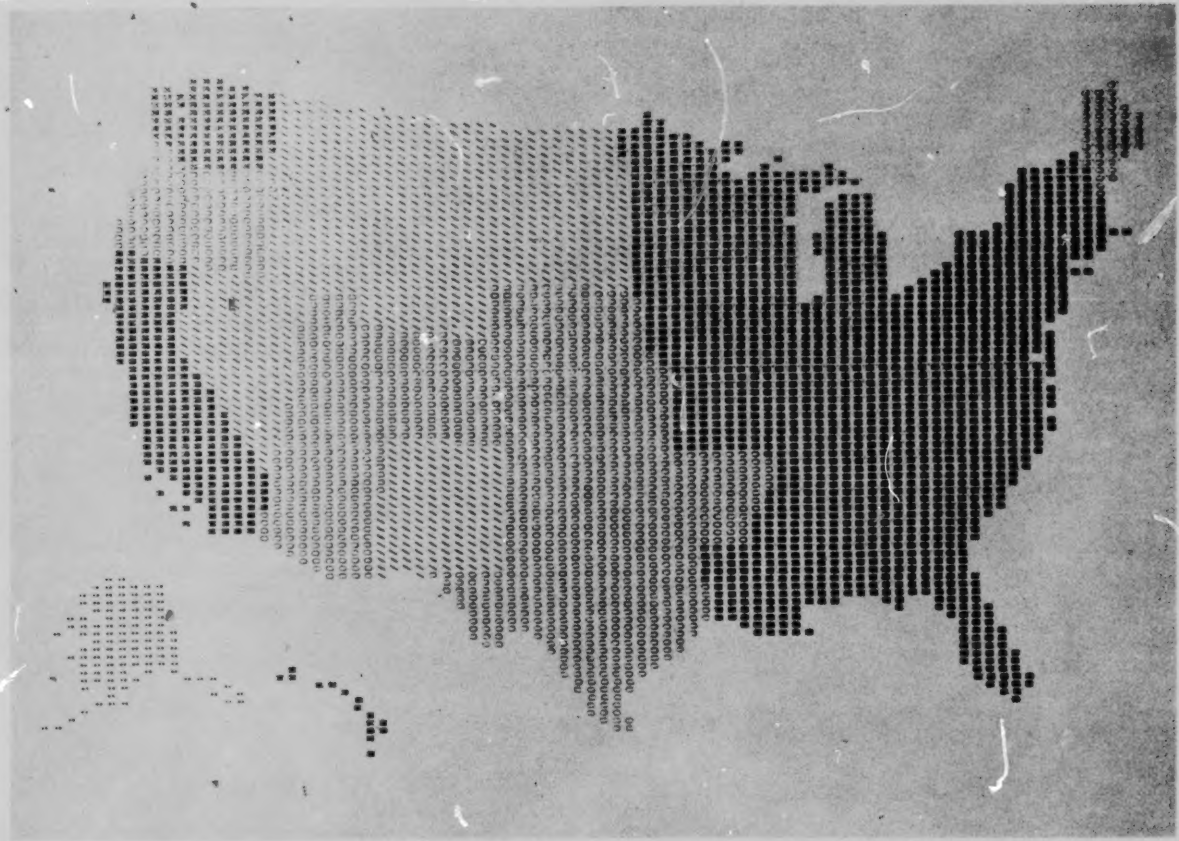


Figure 1. Example of CONMAP output.

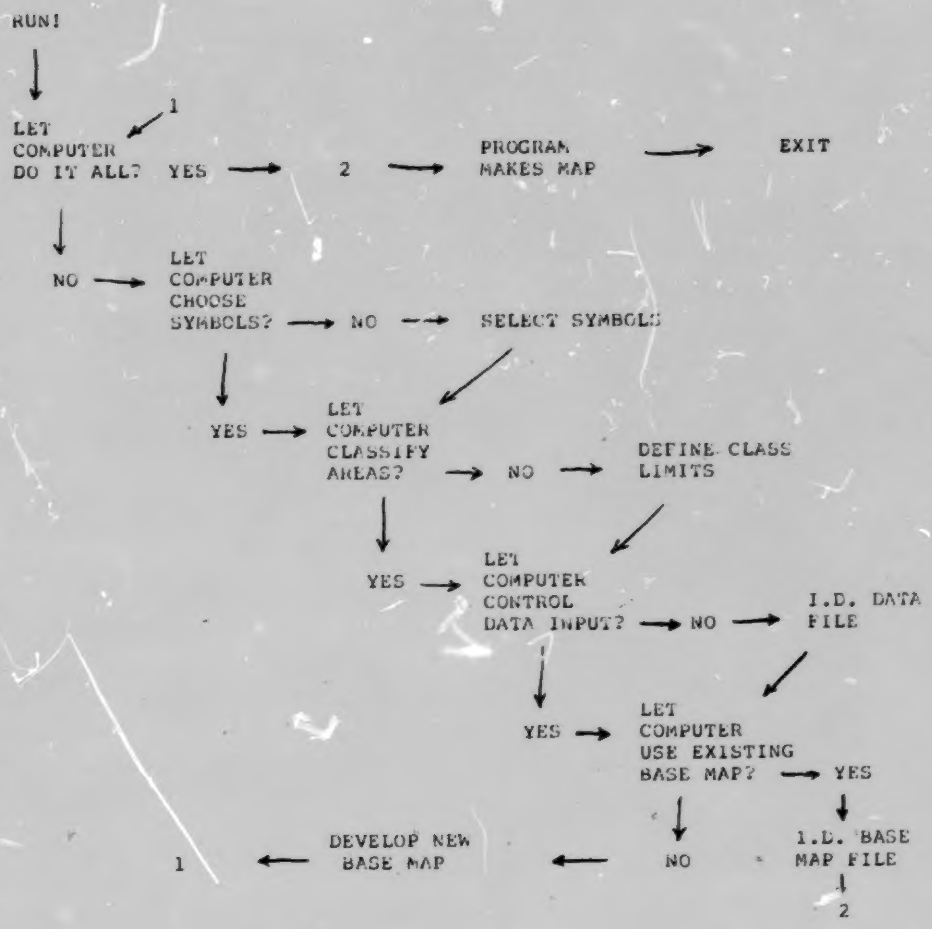
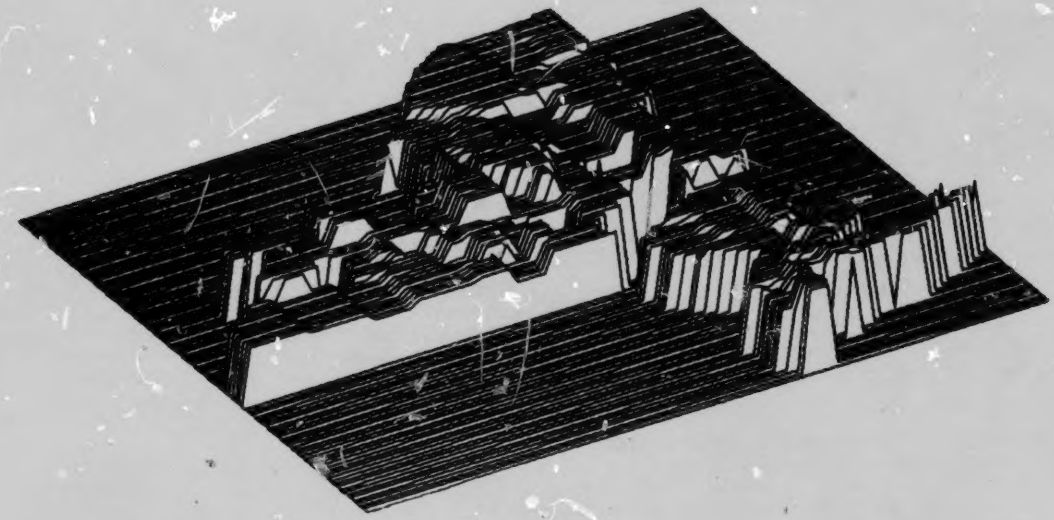

RUN!
1
LET COMPUTER DO IT ALL?
YES
2
PROGRAM MAKES MAP
EXIT
NO
LET COMPUTER CHOOSE SYMBOLS?
NO
SELECT SYMBOLS
YES
LET COMPUTER CLASSIFY AREAS?
NO
DEFINE CLASS LIMITS
YES
LET COMPUTER CONTROL DATA INPUT?
NO
I.D. DATA FILE
YES
LET COMPUTER USE EXISTING BASE MAP?
YES
I.D. BASE MAP FILE
2
NO
DEVELOP NEW BASE MAP
1

Figure 2. Major flow control for CONMAP. Unsophisticated user makes only one decision.



DOELL, R.
MIX EFFECT. 1960-70

Figure 3. Example of PROMAP output.

Authoring Systems for Developing Computer-Based Learning Materials

Chaired by Alfred Bork
University of California, Irvine

Ted Martz, Control Data Corporation

Janene Judd, Brigham Young University

Tryg Ager, SUNY-Binghamton

Alfred Bork, University of California-Irvine

Bobby Brown, The University of Iowa

This session is concerned with the overall process of generating computer-based learning materials in a wide variety of forms. By an authoring system we refer to one particular such strategy. In the early days materials were often developed entirely by a single individual who served many different functions. As we move closer to large-scale production of computer-based learning materials, approaches have been developed which may be more appropriate to a future, sizable industry.

The speakers will describe a variety of authoring systems, talking about which individuals are involved in the work, how each group of individuals works, and how the various groups are connected. Among the important considerations are the effectiveness of the system, the training of the people involved, the use of instructional design, the programming environment, and visual design of both the screen material and of any possible auxiliary media. Different systems will address, to some extent, different aspects of these problems.

CDC Authoring Systems

Ted Martz

Abstract not available at time of printing.

Authoring Procedures for TICCIT

Janene M. Judd, Brigham Young University

Among the many CAI systems, the TICCIT data system is unique: it was designed by instructional scientists rather than programming specialists. One result of the design is that TICCIT does not use a programming language for data entry.

The thrust of this presentation is 1) to briefly acquaint computer users with TICCIT's design and development; 2) to simulate the data entry system; 3) to show other system options available to authors; and 4) to discuss the basic courseware development process with TICCIT at Brigham Young University.

The VOCAL Curriculum Authoring System

Tryg A. Ager, SUNY, Binghamton and Institute for Mathematical Studies in the Social Sciences at Stanford University

VOCAL has the capabilities of traditional test-and-branch CAI as well as answer processing modalities based on research into mathematical reasoning, linguistics, and natural language processing. Additionally, VOCAL is a medium of instructional exposition using both visual and auditory channels of communication.

The VOCAL author can weave visual and audio material together in the presentation phase of a transaction. For each transaction the author selects an answer processor. Answer processors include simple choice questions, drill and practice on language or logical skills, and an interactive proof checker. Answer processors are themselves highly interactive. Once a student response has been processed, the VOCAL author decides what additional interaction occurs.

The design goal of VOCAL is fully interpretive authoring of complex CAI curricula. An author should be able to specify the graphic and audio components of a lecture segment, write questions or problems, then immediately be able to execute the lessons to evaluate both their accuracy and appropriateness.

The Irvine Authoring System

Alfred Bork, University of California, Irvine

The Educational Technology Center at the University of California, Irvine, has had a particular interest in developing effective authoring systems for computer-aided learning materials. Our goals are 1) to allow authors great freedom in specifying materials; 2) to produce highly effective learning material; 3) to produce the material at a reasonable cost; 4) to produce material that can be easily modified on the basis of information gathering during student use; 5) to allow authors full graphic capability; and 6) to aim toward future large-scale production.

Originally work at Irvine proceeded in a time-sharing environment. Based on those experiences we are presently creating a new authoring system that assumes the personal computer as the delivery system. We are not carrying over any of our older software approaches with this new system, but we are using our past experiences as a guide.

We suggest the following steps in such a system:

1. Instructional Design. The critical initial stage is the pedagogical specification of the materials. Authors are not taught any programming language and do not work directly at computers. They work typically in groups of two or three, developing the specifications in an informal flowchart. Pictures are sketched and described.
2. Text Entry. Effective learning material sometimes contains considerable text due to the many possible branches. An experienced typist can enter such material.
3. Screen Design. An important consideration is that of visual design, including both text and graphics. A competent industrial or graphic designer with educational experience is the best person to undertake this task. Special interactive software allowing composition on the screen is essential.
4. Logic. The programmer, working with the products of the screen designer, must add the analysis of student input and the control structures. As these can be elaborate and can on occasion demand considerable calculation, a powerful, higher level language is very desirable. Our current work is with UCSD PASCAL.
5. Initial Review. When the material is first available, the authors, graphic designers, and programmers should run the programs together many times. Extensive changes are often made at this stage.
6. Pretesting. A second revision stage involves using a small group of students selected from the proposed target population. Both computer-saved responses and attitudinal information suggest necessary changes.
7. Modification after Extensive Use. This phase should be repeated several times.

Authoring: A Team Approach

Bobby R. Brown, The University of Iowa

This presentation will focus on coordination of the array of skills and resources required in the design and implementation of computer-based learning materials. The functioning of an authoring team consisting of a content specialist, an instructional designer, and an instructional programmer will be described. Variations in the role and degree of involvement by individual team members will be shown to provide the flexibility required for varying levels of 1) complexity of instructional intent, 2) complexity of implementation, and 3) differential availability of team members.

Computer-Assisted Test Construction via Automatic Program Generation: Using PROBGEN II to Create Individualized Exams and Problem Sets

Ronald W. Collins and Steven J. Duff
Department of Chemistry, Eastern Michigan University
Ypsilanti, Michigan 48197
(313) - 487-0106

Recent advertising by IBM has focused attention on the fact that the per unit cost of computing has dropped dramatically over the past 25 years. For example, computations which cost $1.26 to do on the best available IBM computer in 1952 can be done today for 0.7 of a cent! In other words, computing now costs 180 times less than it did in 1952——actually, it's over 400 times less if the inflationary impact on the buying power of the dollar is considered. This cost reduction is primarily attributable to the great advances in computing hardware, particularly the dramatic improvements in computational speed. Even if the comparisons are made only over the past decade, it can still be said that the cost of computing has decreased significantly.

It seems, however, that the growth and improvements in the field of educational computer usage have not kept pace with the opportunities provided by these favorable hardware and cost changes. Certainly the emergence of minicomputers and microprocessors has enabled more educators to make use of the computer in their teaching; however, from a software viewpoint the changes in instructional computing have not been so impressive. The pedagogical strategies involved and the types of educational computing being used (computer-assisted data reduction, tutorial CAI, computer-assisted test construction, etc.) have not changed significantly. Furthermore, the percentage of all U. S. college and university students who are exposed even once to educational computer usage has not grown as rapidly as predicted by the substantial cost decreases per unit of computing.

This slow growth of educational computing is often blamed primarily on economic factors. Irrespective of how inexpensive computer hardware and computing time become, they still represent add-on costs in an academic institution. As a result, even though educational computing might be a bargain in dollars compared to the actual cost of human instructional time, it is still an extra cost which many institutions may feel they simply cannot afford.

In addition to these economic reasons, software problems are also serious deterrents to the growth of educational computing. Basically, no matter how much hardware is available, and no matter how fast and efficiently it processes data, the typical educator still faces a rather formidable task in developing or acquiring the necessary software (programs, student documentation, etc.) for his/her classroom uses. Personal development of the materials requires either personal programming expertise or funds to pay student programmers. The acquisition of materials from other authors is an alternative approach for any faculty member who has neither the expertise nor the funds to develop his/her own software. Due to the excellent efforts of organizations such as QCPE[1] and CONDUIT[2], this approach is now feasible. High quality, thoroughly tested, well-documented computer-based materials can be acquired at modest cost from these sources with very few machine and language transfer problems. The new user still faces the potential problem though of imbedded pedagogy in the acquired programs; i.e., the programs may use a theory or an equation which the original author favored but which the new instructor has not taught to his/her students.

Another factor which must be considered by any instructor who develops or acquires programs is the question of how long and how often the programs will be used. An extensive investment in time and dollars is probably not warranted if the software will be used only in a low-enrollment course which is taught in alternate years only. Even in high-enrollment classes such as first-year general chemistry, there is still often a problem associated with frequent revisions in the course content. Developing an elegant computer-based supplement for a particular lab experiment can be an expensive short-term investment if curricular modifications result in dropping that experiment from the course within a year or two.

It should also be mentioned that the time required to become proficient in a high-level language like BASIC or FORTRAN has not decreased significantly over the past decade. The changes and improvements in these languages have been minimal, and becoming an efficient, capable programmer is still a major undertaking. Many college instructors either do not have the time or are unwilling to take the time to achieve this programming proficiency.

Considering all these factors, several years ago we decided to maximize the educational impact of our research and development work by concentrating on a computing project that focussed on some fairly universal aspect of the teaching-learning process. To transport our software easily, we decided to write our programs in a standard form of a popular language. Another concern was that subsequent faculty users should need little or no programming expertise. The latter goal is easily achieved with a canned program, but we wanted to write software that would help other instructors create programs and materials tailored to their own course content and pedagogical style.

The most universal component of all courses seems to be giving assignments and exams to the students. Since mastery of any subject often involves repetitive cycles of study and testing, there is also widespread agreement among many college instructors that individualized, repeatable tests are desirable. We thus concentrated our software efforts in the area of computer-assisted test construction (CATC). To meet the criterion of easy transfer, we have written our "PROBGEN" (problem generator) programs in standard FORTRAN.

According to Lippey[3], there are two fundamentally different approaches to CATC. The first is based on the ability of the computer to facilitate information retrieval (CATC-R), and the second involves the computer in a true generation role (CATC-G). In other words, in CATC-R the computer merely manipulates a categorized data base of complete questions with or without answers, but does not actively create or add to the content of the individual questions. This technique is dependent upon the availability of an extensive inventory of questions in machine-readable form (item banking).

In chemistry, the two best known CATC-R systems are those of Moore[4] (based on an earlier system developed by Prosser[5]) and Willis[6]. In each instance there is a main program which controls the processing of the data base of questions. The questions are classified according to topic, degree of difficulty, etc. It is important to stress once again, however, that what goes in, comes out; i.e., the computer does not solve the problems as it sorts through and randomly selects from the available categories. Answers will be printed for the tests that are generated only if the problems had been solved previously and the answers had been included in the data base.

In contrast to this approach, in CATC-G the computational power of the computer is employed to assign values to the variables within problems, and to solve the resultant, unique problems (item generation). In this instance there is no need for an inventory of questions. Instead, one program or subroutine must be written for each type of problem. The author determines the format or skeleton of the desired problem, and then writes a program which will generate values randomly within appropriate limits and fill in the blanks in the problem. Concurrently, the computer solves the problem and provides the correct answer on the key which accompanies the tests generated. In chemistry, the most successful CATC-G system is that developed by Johnson[7] at the University of Pittsburgh, which currently consists of approximately 400 subroutines (each being a question prototype) which are in turn executed and controlled by a main program.

The implementation and expanding use of either type of CATC system can obviously be quite time consuming. In the case of CATC-R, considerable time will be spent either in personally writing new questions or in reviewing and classifying questions obtained elsewhere. On the other hand, in CATC-G the time will be spent in writing new programs, one per problem type.

Assuming programming expertise, the CATC-G approach appears to be a more creative computer-based method. On the other hand, it is obvious that many instructors use CATC-R simply because they do not have the required programming expertise. We therefore concluded that what was really needed was an automatic CATC-G system for non-programmers. In other words, the computer itself should serve as the program/subroutine writer.

Our PROBGEN programs have thus been designed so as to use automatic program generation (APG). The faculty user never (consciously) writes a single line of program code, but he/she is required to abide by the rules of FORTRAN for assigning variable names and for defining arithmetic operations within the desired problem. In interactive mode, PROBGEN prompts the user to provide the generalized solution to each problem, any major restrictions which might apply, the limits on all variables within each problem, and the exact wording for the text of each problem. Following this sequential question-and-answer dialog, PROBGEN then writes or assembles the actual program (a CATC-G module) that will create the desired number of non-identical problems of the defined type. Each test or assignment can consist of one or more problems of as many types as are input to PROBGEN.

In our earlier paper[8] on this APG approach to CATC we described the first prototype of the PROBGEN series (now referred to as PROBGEN I) of programs. Our current paper describes PROBGEN II, a vastly improved version of the earlier software, with greatly expanded capabilities. As was true of the first edition program, the efficient use of PROBGEN II still demands that the faculty user carefully conceptualize and organize each problem into a general format before accessing PROBGEN II online.

In outline form, the steps involved in using PROBGEN II are: (steps 1-9 off-line)

1. Define the type of problem, including the variables (A, B, C,...) whose values will be given, and the variable (X) whose value will be calculated by the student.
2. Combine and arrange the necessary equation(s) to produce an answer equation (s) of the form X = f (A, B, C...).
3. For all non-subscripted variables, A, B, C, ..., specify the upper and lower limits between which single values can be randomly assigned by the program.
4. For all subscripted variables, D(I), E(J), F(K), etc., specify the limits plus the number of values to be randomly assigned.
5. For each variable select a FORTRAN code (variable) name which is consistent with its integer or floating point character.
6. Specify any forbidden values for each variable or forbidden relationships among different variables.
7. Compose the exact wording of the problem as you wish it to appear on the test or problem set.
8. Repeat steps (1) - (7) for each additional problem.
9. Compose the heading for each test and also determine how many problems of each type are to be included per test, and how many tests are desired.
10. (On-line) Access PROBGEN II at a terminal and input the appropriate equation(s), values, and text for each problem type as requested; i.e., respond with the proper input to each question from PROBGEN II. Check all input when it is printed back in order to catch typographical errors before they possibly suppress the execution of the program.
11. Go to the computer center to pick up the tests which are printed on the high-speed line printer.

Although the procedural steps involved in using PROBGEN II are quite similar to those previously used for PROBGEN I, PROBGEN II has greatly expanded technical capabilities compared to the first edition of the program. Table I summarizes the improvements made in PROBGEN II. In order to better illustrate these features, two sample problems have been chosen for a demonstration run of PROBGEN II. The problems are: (1) a strong acid-strong base titration problem, and (2) a combined gas law problem. The concept and format of each problem is given in TABLE II, and the complete user-PROBGEN II dialog is shown in Figure 1, with the user's responses underlined. A sample test consisting of one of each of these two demonstration problems is shown in Figure 2. The answer sheet accompanying the problems as generated by PROBGEN II is shown in Figure 3.

In the processing of problem 1 note that the number of answer equations is specified as 2. This response allows for the branching option; i.e., whether the volume of titrant being considered represents a point in the titration before or after the end point. The restricting condition governing this option——namely whether the mmoles of base added is less than or greater than the initial mmoles of acid——is then input, followed by the additional limitation that the mmoles of base should not be exactly equal to the mmoles of acid. This avoids calculating the pH exactly at the end point. Since the variable name assigned to the volume of base, BVOL(J), is subscripted, the program will randomly assign five different values to this quantity, within the specified limits. The student then must calculate the pH at five different stages in the titration. This type of problem could not have been generated via PROBGEN I since subscripted variables could not be accommodated in that earlier version of this APG-CATC program.

Demonstration problem number 2 involving the combined gas law equation illustrates the use of the PROBGEN II multiple-choice option. Following the input of the single answer equation without branching condition, the user responds yes to the question regarding multiple choice format. Three logical wrong answer equations are then input to the program. These three equations are based on three common student errors: (1) the failure to convert from degrees Kelvin to degrees Celsius; (2) inversion of the pressure factor; and, (3) inversion of the volume factor. Next, integer (I) format is specified for the answers, and the restrictions that $V_1 \neq V_2$ and $P_1 \neq P_2$ are input (this avoids accidental oversimplification of the problem).

Notice that in both demonstration problems inputing the numerical limits for the random range of values for all variables is handled in one compact statement. In the earlier PROBGEN I program this step involved a repetitive series of questions and responses for each individual variable.

For the sake of brevity just two sample problems have been used to illustrate the capabilities of PROBGEN II. Although not shown, PROBGEN II can handle up to twenty variables within a given problem. Furthermore, although the two examples used in this paper were both chemistry problems, it should be obvious that PROBGEN II could be applied in other disciplines as well. Any numerical problem which can be systematically generalized into an answer equation may be processed via PROBGEN II, irrespective of whether the topic is physics, biology, or economics.

PROBGEN II is applicable only to the generation of quantitative problems. It cannot be used to create theory or discussion questions, and thus PROBGEN II cannot produce as wide a range of test types as can a CATC-R system.

PROBGEN II can be employed to create new problems by a faculty member who is a non-programmer, whereas additions to the Johnson[7] CATC-G system require considerable programming expertise. Thus, PROBGEN II not only generates individualized, non-identical problem sets or tests for student use, but also permits the faculty member to easily and quickly (usually in 15-20 minutes) individualize the style of the problem according to his/her pedagogical preferences.

No attempt has been made in this paper to justify the use of individualized assignments or tests, or to describe the proper logistics for their most effective use. Instead, the focus has been on the design and use of PROBGEN II, and the reader must decide whether he/she wishes to make use of such materials in his/her teaching.

In conclusion, PROBGEN II represents a significant breakthrough in making CATC technology available to the non-programmer. Although it does not allow the user to communicate completely free of programming syntax, PROBGEN II is an efficient APG system which relies on the computer to write the programs involved. For more information on the state of the art in the field of APG, consult the excellent review article by Prywes.[9] In terms of overall usage pattern, PROBGEN II can be employed as a stand-alone CATC-G system (Figure 4) or as a generator of numerous non-identical problems which can subsequently be used as the data base for a CATC-R System (Figure 5). The present FORTRAN version of PROBGEN II is available upon request (on magnetic tape) for a modest fee to cover the handling and mailing costs.

References

1. Quantum Chemistry Program Exchange (QCPE), Department of Chemistry, Indiana University, Bloomington, IN 47401

2. CONDUIT, P. O. Box 388, Iowa City, Iowa 52240

3. Computer-Assisted Test Construction, G. Lippey, Editor, Educational Technology Publications, Englewood Cliffs, NJ 07632 (1974).

4. "Computer-Generated Repeatable Tests in Chemistry", J. W. Moore, Proceedings of the Conference on Computers in Chemical Education and Research, Northern Illinois University (1971).

5. "Repeatable Tests", F. Prosser, Educational Technology, 13 (3), 34 (1973)

6. "The SOCRATES CATC System at the California State Universities and Colleges", W. V. Willis and O. Seely, paper presented at the First Chemical Congress of the North American Continent, Mexico City, Mexico (1975).

7. "Pittsburgh's Computer-Generated Repeatable Exam System", K. J. Johnson, Proceedings of the 1973 Conference on Computers in the Undergraduate Curricula, p. 199 (1973).

8. "PROBGEN: An Interactive Computer Program for Faculty Use in Creating and Generating Individualized, Non-identical Problem Sets and Tests," Proceedings of the Ninth Conference on Computers in the Undergraduate Curricula, p. 100 (1978).

9. "Automatic Generation of Computer Programs," N. S. Prywes, Advances in Computers, Volume 16, M. Rubinoff and M. C. Yovits, Editors; pp. 57-125, Academic Press, NY (1977).

PROBGEN I	vs.	PROBGEN II
(1) Variables not actually needed to solve the problem must nevertheless be included twice in the answer equation as "dummy variables" (in canceling positions)——otherwise numerical values cannot be assigned to such variables (2) Each variable can have either a single, constant value in a given problem or it can be randomized——demands lengthy, repetitive text in the program (3) No subscripted variables can be used; only single-value variables within each problem (4) No user control over significant figures	VARIABLES - type, user control, etc.	(1) "Dummy variables" not needed in answer equations; the program calls for the limits on such variables in a separate step (2) All variables are assumed to be randomized; single value quantities are to be included as numerical constants in the answer equation. Program is less text intensive. (3) Subscripted variables of three types may be used: X(I)-program assigns n regularly-spaced incrementa values within specified limits X(J)-program assigns n random values within specified limits X(K)-program accepts n specific values for the variable (4) User can dictate significant figures for each variable via the form of the input for the limits
(1) All problems must lead to a single answer; i.e., no problems with several parts, each calling for an answer (2) All problems are printed in "open style"; i.e, multiple choice format is not possible	PROBLEM FORMATS - several parts, multiple choice	(1) Problems may have several parts, with the parts being either inter-related or completely independe of each other. If related, the answer equations may be set up sequentially; i.e., using one answer to calculate the next, etc. (2) The user has the option to use multiple choice format for any problem. In addition to the correct answer, the program provides three wrong answers via: ___calculations using three "logical wrong answer equations" input by the user, or ___purely random selection of three other values within a range specified by the user
(1) All solutions must be based on the self-contained answer equation provided by the user	ARITHMETIC CAPABILITIES	(1) Solutions may involve the use of standard library subroutines; e.g., if the answer equation is a cubic, the user may call a Newton-Raphson subroutine for the solution

TABLE I Comparison of PROBGEN I and PROBGEN II

STRONG ACID-STRONG BASE TITRATION PROBLEM		GAS LAW PROBLEM
A titration problem involving five successive additions of the titrant, NaOH, to a solution of HCl	General Concept of the Problem	Combined gas law problem using $\frac{P_1V_1}{T_1} = \frac{P_2V_2}{T_2}$
The initial volume, ml_A, of acid The concentration of the acid, M_A The concentration of the base, M_B Five total volumes of base added, ml_{B_1}, ml_{B_2}, ... etc,	Given (or to be assigned by the program)	All initial values, P_1, V_1, and T_1, The final values for P_2 and V_2
Five pH values; one for each total volume of base added	To be calculated by the student	The final temperature, t_2 in degrees Celsius
(i) if $(ml_A \times M_A) > (ml_{B_n} \times M_B)$ $pH = -\log \left[\frac{(ml_A \times M_A) - (ml_{B_n} \times M_B)}{ml_A + ml_B} \right]$ (ii) if $(ml_A \times M_A) < (ml_{B_n} \times M_B)$ $pH = 14.00 + \log \left[\frac{(ml_{B_n} \times M_B) - (ml_A \times M_A)}{ml_A + ml_B} \right]$	Answer equation(s) and the special condition for each (if any).	$t_2 = \left[T_1 \times \frac{P_2}{P_1} \times \frac{V_2}{V_1} \right] - 273$ $P_2 \neq P_1$ $V_2 \neq V_1$

TABLE II General Format of the Two Problems Used to Demonstrate PROBGEN II

Figure 1:
User-PROBGEN II
Dialog

```
.,RUN PROBII

********************
********************

PROBGEN
EASTERN MICHIGAN UNIVERSITY
DEPARTMENT OF CHEMISTRY
WRITTEN BY:
   DR. RONALD COLLINS
   STEVE J. DUFF

********************
********************

FOR PURPOSES OF RECORD KEEPING
PLEASE TYPE IN SOME USER ID INFORMATION
      NAME
      LOCATION
      PURPOSE (DEMO OR PRODUCTION,ETC)
*****  PROBGEN II  DEMONSTRATION  *****

********************
*                  *
* START OF PROGRAM *
*                  *
********************


HOW MANY PROBLEMS ARE TO BE INCLUDED ON EACH
ASSIGNMENT OR TEST? :2

WOULD YOU LIKE TO SEE THE
INSTRUCTIONS FOR TYPING IN THE EQUATIONS FOR EACH
PROBLEM? (YES OR NO)>NO

************************
*                      *
* PROCESSING PROBLEM 1 *
*                      *
************************

HOW MANY DIFFERENT ANSWER EQUATIONS WILL BE
USED IN PROBLEM  1 (TYPE "SAME" .. IF YOU WANT ANOTHER
PROBLEM OF THE SAME TYPE AS THE PREVIOUS PROBLEM)? :2

TYPE ANSWER EQUATION   1:@PH(J)=-@ALOG10(((AVOL*ACONC)-(BVOL(J)*BCONC))/(AVOL*B

HERE IS THE EQUATION AS YOU TYPED IT:@PH(J)=-@ALOG10(((AVOL*ACONC)-(BVOL(J)*BCO

IS ANYTHING WRONG WITH THIS EQUATION?
TYPE YES OR NO >NO

IS THERE A CONDITION RESTRICTING THE USE OF
THIS EQUATION? (TYPE YES OR NO) :YES

TYPE A VARIABLE, ALGEBRAIC EXPRESSION,
OR CONSTANT VALUE (FIRST PART OF THE CONDITIONAL STATEMENT)
PLEASE TYPE>(AVOL*ACONC)

THIS IS WHAT WAS TYPED. IS IT CORRECT?
(TYPE YES OR NO)   :(AVOL*ACONC)
YES
```

Figure 1:
page 2-continued
User-PROBGEN II Dialog

```
TYPE A VARIABLE, ALGEBRAIC EXPRESSION,
OR CONSTANT VALUE (SECOND PART OF THE CONDITIONAL STATEMENT)
PLEASE TYPE>(BVOL(J)*BCONC)

THIS IS WHAT WAS TYPED, IS IT CORRECT?
(TYPE YES OR NO)  :(BVOL(J)*BCONC)
YES

TYPE THE CONDITIONAL OPERATOR
(FIRST PART OF THE STATEMENT AS IT MUST RELATE
TO THE SECOND PART)
    EXAMPLES:
         .LT.   :  LESS THAN
         .GT.   :  GREATER THAN
         .EQ.   :  EQUAL
         .GE.   :  GREATER THAN AND EQUAL

    PLEASE TYPE(.EQ.,.LT.,....ETC):.GT.

ARE YOU SURE THAT .GT. IS CORRECT?
(TYPE YES OR NO) :YES

TYPE ANSWER EQUATION   2:@PH(J)=14.00+@ALCG10(((BVOL(J)*BCONC)-(AVOL*ACONC))/(A

HERE IS THE EQUATION AS YOU TYPED IT:@PH(J)=14.00+@ALOG10(((BVOL(J)*BCONC)-(AVOL

IS ANYTHING WRONG WITH THIS EQUATION?
TYPE YES OR NO >NO

IS THERE A CONDITION RESTRICTING THE USE OF
THIS EQUATION? (TYPE YES OR NO) :YES

TYPE A VARIABLE, ALGEBRAIC EXPRESSION,
OR CONSTANT VALUE (FIRST PART OF THE CONDITIONAL STATEMENT)
PLEASE TYPE>(AVOL*ACONC)

THIS IS WHAT WAS TYPED, IS IT CORRECT?
(TYPE YES OR NO)  :(AVOL*ACONC)
YES

TYPE A VARIABLE, ALGEBRAIC EXPRESSION,
OR CONSTANT VALUE (SECOND PART OF THE CONDITIONAL STATEMENT)
PLEASE TYPE>(BVOL(J)*BCONC)

THIS IS WHAT WAS TYPED, IS IT CORRECT?
(TYPE YES OR NO)  :(BVOL(J)*BCONC)
YES

TYPE THE CONDITIONAL OPERATOR
(FIRST PART OF THE STATEMENT AS IT MUST RELATE
TO THE SECOND PART)
    EXAMPLES:
         .LT.   :  LESS THAN
         .GT.   :  GREATER THAN
         .EQ.   :  EQUAL
         .GE.   :  GREATER THAN AND EQUAL

    PLEASE TYPE(.EQ.,.LT.,....ETC):.LT.

ARE YOU SURE THAT .LT. IS CORRECT?
(TYPE YES OR NO) :YES

WILL THE PROBLEM BE SET UP IN MULTIPLE CHOICE FORMAT?
REMEMBER THAT SUBSCRIPTED VARIABLES CANNOT BE USED
WITH MULTIPLE CHOICE, AND THE TOTAL NUMBER OF ANSWERS
LISTED WILL BE FOUR, INCLUDING THE CORRECT ANSWER. >NO

ARE THERE ANY VARIABLES THAT YOU WANT TO
INCLUDE IN THE PROBLEM, BUT WHICH DO NOT APPEAR IN
ANSWER EQUATIONS? (TYPE YES OR NO) :NO
```

Figure 1:
page 3-continued

User-PROBGEN II
Dialog

```
TYPE THE APPROPRIATE LIMITS (LOWER LIMIT,
UPPER LIMIT) OR SPECIFIC VALUES FOR EACH OF
THE FOLLOWING VARIABLES TAKEN FROM YOUR EQUATIONS!

          AVOL        >20.00,30.00

          ACONC       >0.1000,0.2000

          BVOL(J)     >5.00,80.00,5

          BCONC       >0.1000,0.2000

ARE THERE ANY NUMERICAL LIMITATIONS OR
FORBIDDEN VALUES FOR ANY VARIABLES, ALONE OR IN COMBINATION?
(TYPE YES OR NO) :YES

WHAT IS THE VARIABLE, ALEGRAIC EXPRESSION,
OR CONSTANT VALUE (FIRST PART OF CONDITIONAL STATEMENT)
PLEASE TYPE :(AVOL*ACONC)

THIS IS WHAT WAS TYPED IN, IS IT CORRECT?(AVOL*ACONC)
(TYPE YES OR NO) :YES

TYPE ".LT.",".GT.",".LE.",".GE.",".EQ.",".NE." IN
HOW FIRST STATEMENT) WILL RELATE TO THE SECOND STATEMENT)  :.NE.

TYPE A VARIABLE, ALGEBRAIC EXPRESSION,
OR CONSTANT VALUE (SECOND PART OF CONDITIONAL STATEMENT)
PLEASE TYPE :(BVOL(J)*BCONC)

THIS IS WHAT WAS TYPED IN, IS IT CORRECT?(BVOL(J)*BCONC)
(TYPE YES OR NO) :YES

ARE THERE ANY MORE LIMITATIONS OF THIS TYPE :NO

HOW MANY LINES OF TYPING WILL THE TEXT
OF THE PROBLEM REQUIRE (MAXIMUM:10) :6


TYPE THE TEXT OF THE PROBLEM ...
    KEEP IN MIND:
        1.  VARIABLE OCCURRENCES(TYPE THE SIGN "@" FOLLOWED BY THE VARIABLE NAME)
        2.  HIT THE RETURN KEY AFTER EACH LINE OF TEXT
        3.  TYPE IN AS MANY LINES AS REQUESTED EVEN IF
            YOU HAVE TO TYPE BLANK LINES AT THE END

    PLEASE TYPE:

   (DEMONSTRATION OF THE USE OF SUBSCRIPTED VARIABLES)
ASSUME THAT YOU ARE TITRATING A @AVOL - ML SAMPLE OF
@ACONC M HCL WITH @BCONC M NAOH.  CALCULATE THE PH OF THE
SOLUTION AFTER THE ADDITION OF EACH OF THE FOLLOWING TOTAL
VOLUMES (ML) OF NAOH:  @BVOL(J).

HERE IS YOUR TEXT (CHECK IT VERY CAREFULLY)
IF THERE ARE ANY ERRORS, YOU CAN RE-TYPE THE
TEXT BELOW:
   (DEMONSTRATION OF THE USE OF SUBSCRIPTED VARIABLES)
ASSUME THAT YOU ARE TITRATING A @AVOL - ML SAMPLE OF
@ACONC M HCL WITH @BCONC M NAOH.  CALCULATE THE PH OF THE
SOLUTION AFTER THE ADDITION OF EACH OF THE FOLLOWING TOTAL
VOLUMES (ML) OF NAOH:  @BVOL(J).

ARE THERE ANY ERRORS IN YOUR TEXT? (TYPE YES OR NO) :NO
```

Figure 1:
page 4-continued
User-PROBGEN II.
Dialog

```
***************************
*                         *
*  PROCESSING PROBLEM  2  *
*                         *
***************************

HOW MANY DIFFERENT ANSWER EQUATIONS WILL EE
USED IN PROBLEM  2 (TYPE "SAME" . IF YOU WANT ANOTHER
PROBLEM OF THE SAME TYPE AS THE PREVIOUS PROBLEM)? !1

TYPE ANSWER EQUATION  1:@NTTWO=(NPTWO*NVTWO*(NTONE+273.))/((7.6E5)*NPONE*NVONE;

HERE IS THE EQUATION AS YOU TYPED IT:@NTTWO=(NPTWO*NVTWO*(NTONE+273.))/((7.6E5)*

IS ANYTHING WRONG WITH THIS EQUATION?
TYPE YES OR NO >NO

IS THERE A CONDITION RESTRICTING THE USE OF
THIS EQUATION? (TYPE YES OR NO) :NO

WILL THE PROBLEM BE SET UP IN MULTIPLE CHOICE FORMAT?
REMEMBER THAT SUBSCRIPTED VARIABLES CANNOT BE USED
WITH MULTIPLE CHOICE, AND THE TOTAL NUMBER OF ANSWERS
LISTED WILL BE FOUR, INCLUDING THE CORRECT ANSWER, >YES

TYPE THE THREE ADDITIONAL EQUATIONS LEADING TO THE WRONG ANSWERS :


WRONG1=(NPTWO*NVTWO*(NTONE+273.))/((7.6E5)*NPONE*NVONE)

WRONG2=(NPONE*(NVTWO/1000.)*(NTONE+273.))/((NPTWO/760.)*NVONE)-273.

WRONG3=((NPTWO/760.)*NVONE*(NTONE+273.))/(NPONE*(NVTWO/1000.))-273.

SHOULD THE ANSWER CHOICES BE TYPED IN
INTEGER, FLOATING POINT, OR EXPONENTIAL FORMAT
(TYPE I, F, OR E, RESPECTIVELY)?I

ARE THERE ANY VARIABLES THAT YOU WANT TO
INCLUDE IN THE PROBLEM, BUT WHICH DO NOT APPEAR IN
ANSWER EQUATIONS? (TYPE YES OR NO) :YES

TYPE ALL SUCH "DUMMY VARIABLES" THAT DO NOT
APPEAR IN THE EQUATIONS (SEPARATED BY A COMMAS):GRAMS

TYPE THE APPROPRIATE LIMITS (LOWER LIMIT,
UPPER LIMIT) OR SPECIFIC VALUES FOR EACH OF
THE FOLLOWING VARIABLES TAKEN FROM YOUR EQUATIONS:


          NPTWO      >3000,7000

          NVTWO      >10000,80000

          NTONE      >5,75

          NPONE      >2,10

          NVONE      >10,100

          GRAMS      >50.00,100.00

ARE THERE ANY NUMERICAL LIMITATIONS OR
FORBIDDEN VALUES FOR ANY VARIABLES, ALONE OR IN COMBINATION?
(TYPE YES OR NO) :YES

WHAT IS THE VARIABLE, ALEGRAIC EXPRESSION,
OR CONSTANT VALUE (FIRST PART OF CONDITIONAL STATEMENT)
PLEASE TYPE :NPONE

THIS IS WHAT WAS TYPED IN, IS IT CORRECT?NPONE
(TYPE YES OR NO) :YES
```

Figure 1:
Page 5-continued
User-PROBGEN II
Dialog

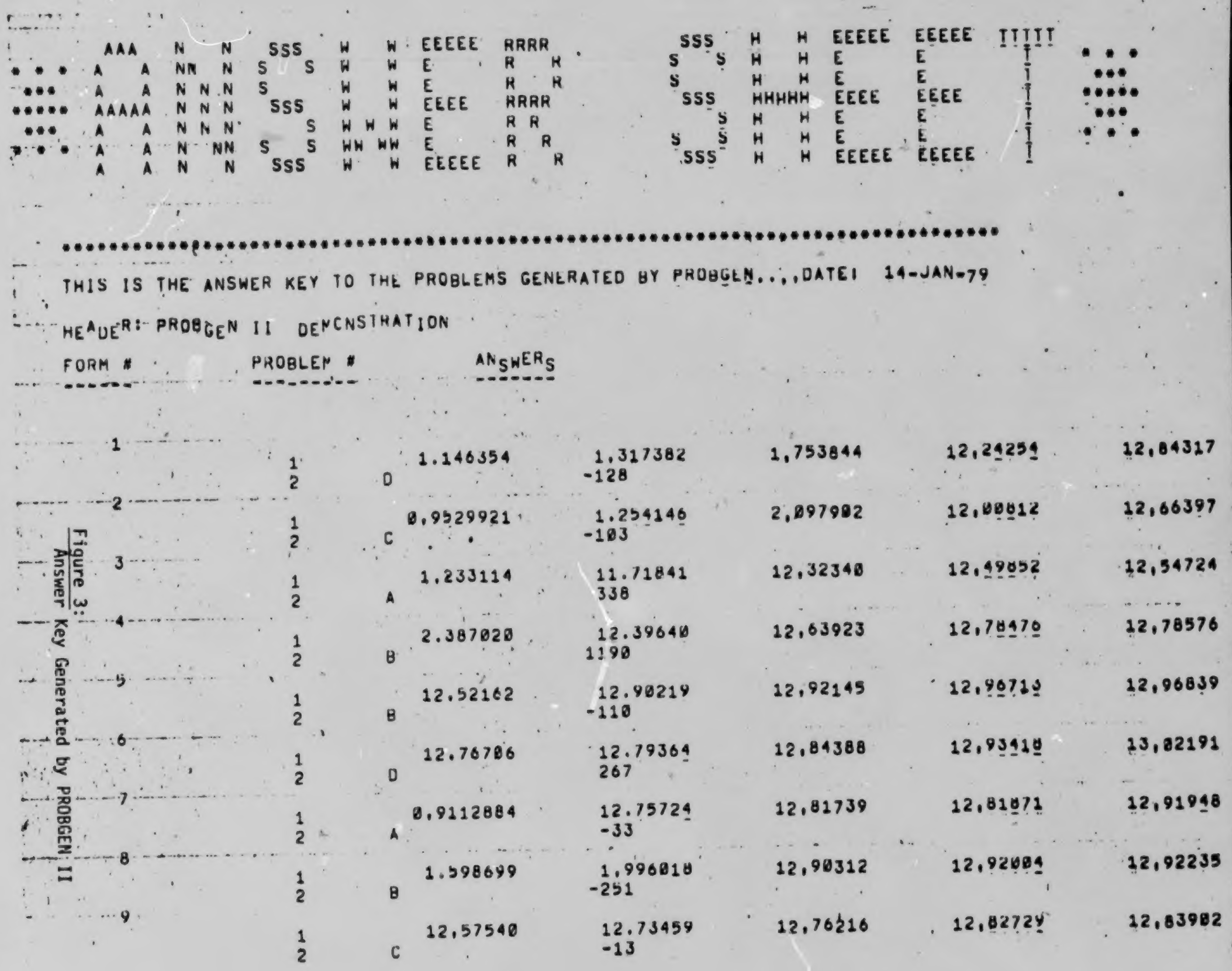
```
TYPE ".LT.",".GT.",".LE.",".GE.",".EQ.",".NE." IN
HOW FIRST STATEMENT) WILL RELATE TO THE SECOND STATEMENT)  :.NE.

TYPE A VARIABLE, ALGEBRAIC EXPRESSION,
OR CONSTANT VALUE (SECOND PART OF CONDITIONAL STATEMENT)
PLEASE TYPE :(NPTWO/760)

THIS IS WHAT WAS TYPED IN, IS IT CORRECT?(NPTWO/760)
(TYPE YES OR NO) :YES

ARE THERE ANY MORE LIMITATIONS OF THIS TYPE :YES

WHAT IS THE VARIABLE, ALEGRAIC EXPRESSION,
OR CONSTANT VALUE (FIRST PART OF CONDITIONAL STATEMENT)
PLEASE TYPE :NVONE

THIS IS WHAT WAS TYPED IN, IS IT CORRECT?NVONE
(TYPE YES OR NO) :YES

TYPE ".LT.",".GT.",".LE.",".GE.",".EQ.",".NE." IN
HOW FIRST STATEMENT) WILL RELATE TO THE SECOND STATEMENT)  :.NE.

TYPE A VARIABLE, ALGEBRAIC EXPRESSION,
OR CONSTANT VALUE (SECOND PART OF CONDITIONAL STATEMENT)
PLEASE TYPE :(NVTWO/1000)

THIS IS WHAT WAS TYPED IN, IS IT CORRECT?(NVTWO/1000)
(TYPE YES OR NO) :YES

ARE THERE ANY MORE LIMITATIONS OF THIS TYPE :NO

HOW MANY LINES OF TYPING WILL THE TEXT
OF THE PROBLEM REQUIRE (MAXIMUM:10) :8

TYPE THE TEXT OF THE PROBLEM ...
    KEEP IN MIND:
       1.  VARIABLE OCCURRENCES(TYPE THE SIGN "#" FOLLOWED BY THE VARIABLE NAME)
       2.  HIT THE RETURN KEY AFTER EACH LINE OF TEXT
       3.  TYPE IN AS MANY LINES AS REQUESTED EVEN IF
           YOU HAVE TO TYPE BLANK LINES AT THE END

     PLEASE TYPE:

   (DEMONSTRATION OF MULTIPLE CHOICE FORMAT)
ASSUME THAT YOU HAVE EXACTLY #GRAMS GRAMS CF NITROGEN
GAS WHICH OCCUPIES #NVONE LITERS AND EXERTS A PRESSURE
OF #NPONE ATMOSPHERES AT #NTONE DEGREES CELSIUS.  WHAT
WILL THE TEMPERATURE (IN DEGREES CELSIUS) CF THE GAS BE
IF THE VOLUME CHANGES TO #NVTWO MILLILITERS AND THE
PRESSURE TO #NPTWO TORR?  (CIRCLE THE CORRECT ANSWER.)


HERE IS YOUR TEXT (CHECK IT VERY CAREFULLY)
IF THERE ARE ANY ERRORS, YOU CAN RE-TYPE THE
TEXT BELOW:
   (DEMONSTRATION OF MULTIPLE CHOICE FORMAT)
ASSUME THAT YOU HAVE EXACTLY #GRAMS GRAMS OF NITROGEN
GAS WHICH OCCUPIES #NVONE LITERS AND EXERTS A PRESSURE
OF #NPONE ATMOSPHERES AT #NTONE DEGREES CELSIUS.  WHAT
WILL THE TEMPERATURE (IN DEGREES CELSIUS) CF THE GAS BE
IF THE VOLUME CHANGES TO #NVTWO MILLILITERS AND THE
PRESSURE TO #NPTWO TORR?  (CIRCLE THE CORRECT ANSWER.)

ARE THERE ANY ERRORS IN YOUR TEXT? (TYPE YES OR NO) :NO

FORTRAN:  QUEST.F4
FORTRAN:  HEADER.F4
LINK:   LOADING
[LNKXCT QUEST EXECUTION]
```

```
PROBGEN II  DEMONSTRATION                            FORM: 7  DATE: 19-JAN-79

PPPP   RRRR    OOO   BBBB    GGG   EEEEE  N   N          III   III
P   P  R   R  O   O  B   B  G   G  E      NN  N           I     I
P   P  R   R  O   O  B   B  G      E      N N N           I     I
PPPP   RRRR   O   O  BBBB   G  GG  EEEE   N N N           I     I
P      R R    O   O  B   B  G   G  E      N N N           I     I
P      R  R   O   O  B   B  G   G  E      N  NN           I     I
P      R   R   OOO   BBBB    GGG   EEEEE  N   N          III   III
 1.     (DEMONSTRATION OF THE USE OF SUBSCRIPTED VARIABLES)
    ASSUME THAT YOU ARE TITRATING A 29.98 - ML SAMPLE OF
     .1965 M HCL WITH  .1900 M NAOH.  CALCULATE THE PH OF THE
    SOLUTION AFTER THE ADDITION OF EACH OF THE FOLLOWING TOTAL
    VOLUMES (ML) OF NAOH:   7.28, 57.26, 63.22, 63.37, 78.39.









 2.     (DEMONSTRATION OF MULTIPLE CHOICE FORMAT)
    ASSUME THAT YOU HAVE EXACTLY  92.80 GRAMS OF NITROGEN
    GAS WHICH OCCUPIES  49 LITERS AND EXERTS A PRESSURE
    OF  9 ATMOSPHERES AT 12 DEGREES CELSIUS.  WHAT
    WILL THE TEMPERATURE (IN DEGREES CELSIUS) OF THE GAS BE
    IF THE VOLUME CHANGES TO 58598 MILLILITERS AND THE
    PRESSURE TO 4804 TORR?  (CIRCLE THE CORRECT ANSWER.)

              A.   -33   C.   239
              B.   212   D.  -105
```

Figure 2:
Sample Test/Problem Set Generated
by PROBGEN II

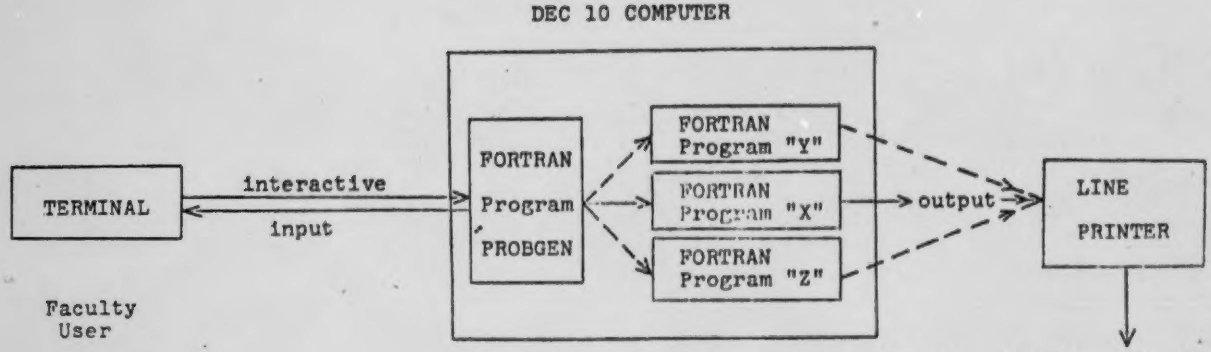
```
                                                   FORM: 8  DATE: 14-JAN-79
PROBGEN II  DEMONSTRATION

PPPP   RRRR    OOO   BBBB    GGG   EEEEE  N   N         III   III
P   P  R   R  O   O  B   B  G   G  E      NN  N          I     I
P   P  R   R  O   O  B   B  G      E      N N N          I     I
PPPP   RRRR   O   O  BBBB   G  GG  EEEE   N N N          I     I
P      R R    O   O  B   B  G   G  E      N N N          I     I
P      R  R   O   O  B   B  G   G  E      N  NN          I     I
P      R   R   OOO   BBBB    GGG   EEEEE  N   N         III   III

 1.     (DEMONSTRATION OF THE USE OF SUBSCRIPTED VARIABLES)
    ASSUME THAT YOU ARE TITRATING A 27.69 - ML SAMPLE OF
     .1265 M HCL WITH  .1676 M NAOH.  CALCULATE THE PH OF THE
    SOLUTION AFTER THE ADDITION OF EACH OF THE FOLLOWING TOTAL
    VOLUMES (ML) OF NAOH:  14.55, 18.14, 65.28, 68.78, 69.29.




 2.     (DEMONSTRATION OF MULTIPLE CHOICE FORMAT)
    ASSUME THAT YOU HAVE EXACTLY  78.75 GRAMS OF NITROGEN
    GAS WHICH OCCUPIES  67 LITERS AND EXERTS A PRESSURE
    OF  9 ATMOSPHERES AT 10 DEGREES CELSIUS.  WHAT
    WILL THE TEMPERATURE (IN DEGREES CELSIUS) OF THE GAS BE
    IF THE VOLUME CHANGES TO 11100 MILLILITERS AND THE
    PRESSURE TO 3077 TORR?  (CIRCLE THE CORRECT ANSWER.)

                  A.    21   C.  -168
                  B.  -251   D.   495
```

Figure 2: page 2 - continued
Sample Test/Problem Set Generated
by PROBGEN II



```
        AAA   N   N  SSS  W   W EEEEE RRRR        SSS  H   H EEEEE EEEEE TTTTT
* * *  A   A  NN  N S   S W   W E     R   R      S   S H   H E     E       T     * * *
 ***   A   A  N N N S     W   W E     R   R      S     H   H E     E       T      ***
*****  AAAAA  N N N  SSS  W   W EEEE  RRRR        SSS  HHHHH EEEE  EEEE    T     *****
 ***   A   A  N  NN     S W W W E     R R            S H   H E     E       T      ***
* * *  A   A  N  NN S   S WW WW E     R  R       S   S H   H E     E       T     * * *
       A   A  N   N  SSS  W   W EEEEE R   R       SSS  H   H EEEEE EEEEE   T

    ******************************************************************************

    THIS IS THE ANSWER KEY TO THE PROBLEMS GENERATED BY PROBGEN...,DATE:  14-JAN-79

    HEADER: PROBGEN II  DEMONSTRATION

    FORM #            PROBLEM #          ANSWERS
    ------            ---------          -------

        1
                        1         1.146354      1,317382      1,753844      12,24254      12,84317
                        2      D              -128
        2
                        1        0,9529921      1.254146      2,097902      12,00812      12,66397
                        2      C              -103
        3
                        1         1,233114      11.71841      12,32340      12,49852      12,54724
                        2      A               338
        4
                        1         2.387020      12.39640      12,63923      12,78476      12,78576
                        2      B              1190
        5
                        1         12.52162      12.90219      12,92145      12,96713      12,96839
                        2      B              -110
        6
                        1         12.76706      12.79364      12,84388      12,93410      13,02191
                        2      D               267
        7
                        1        0,9112884      12.75724      12,81739      12,81871      12,91948
                        2      A               -33
        8
                        1         1.598699      1,996010      12,90312      12,92004      12,92235
                        2      B              -251
        9
                        1         12,57540      12.73459      12,76216      12,82729      12,83902
                        2      C               -13
```

Figure 3: Answer Key Generated by PROBGEN II


DEC 10 COMPUTER
TERMINAL
interactive
input
Faculty User
FORTRAN Program PROBGEN
FORTRAN Program "Y"
FORTRAN Program "X"
FORTRAN Program "Z"
output
LINE PRINTER

One or more programs are created by PROBGEN, one for each problem type X, Y, and/or Z as specified in the input.

Batch production of n tests or problem sets with accompanying answer key; each test consists in turn of one or more problems of each type, X, Y, and/or Z.

FIGURE 4 Outline of the Use and Operation of program PROBGEN at Eastern Michigan University

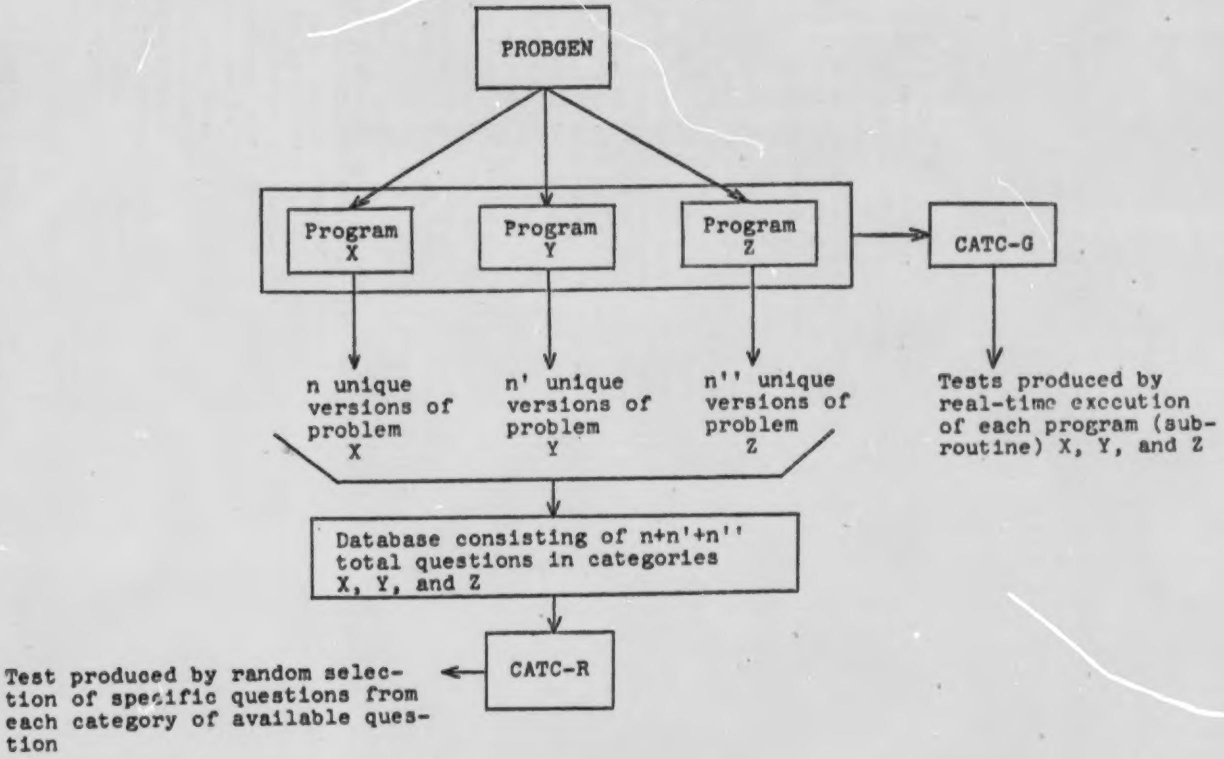

PROBGEN
Program X
Program Y
Program Z
CATC-G
n unique versions of problem X
n' unique versions of problem Y
n'' unique versions of problem Z
Tests produced by real-time execution of each program (subroutine) X, Y, and Z
Database consisting of n+n'+n'' total questions in categories X, Y, and Z
CATC-R
Test produced by random selection of specific questions from each category of available question

FIGURE 5 Use of PROBGEN in conjunction with either CATC-R or CATC-G Systems

COMPUTER DIALOGS TO AID FORMAL REASONING

Arnold Arons, Physics Department, University of Washington, Seattle, WA 98105
Alfred Bork, Physics Department, University of California, Irvine, CA 92717

Abstract

This paper describes a group of six computer dialogs developed jointly by two authors that principally assist students in making the transition from concrete to formal reasoning.

Effective teaching and learning at any level depends on appreciation and understanding the patterns of intellectual growth in students; as students progress from early childhood to maturity, intellectual growth and development takes place. Even casual observations will show that intellectual skills possible for many adults are not possible for a young child.

One of the serious students of such development conducting many empirical studies is the French-Swiss psychologist, Jean Piaget. Piaget, based on his studies of children, outlined a series of stages leading to the fully mature student who is capable of formal or abstract reasoning.

These ideas have influenced important curriculum development projects in the United States such as the Science Curriculum Improvement Study conducted under the direction of Robert Karplus at the Lawrence Hall of Science, University of California, Berkeley.

It has only been within the past few years that these concepts have been applied to university education. Piaget represented the final transition from concrete forms of reasoning to formal forms of reasoning as occurring at about the ages of fourteen or fifteen. Recently a number of surveys examined American college students and determined, much to the surprise of many, that approximately one-half of all college students are not capable of all aspects of reasoning. Furthermore, this ratio does not seem to change during the four years of university life.

This failure to attain full reasoning capability presents two problems. One is the immediate problem of the person teaching college courses. Most college courses assume the attributes of formal reasoning. It is, therefore, not surprising that many students have considerable difficulty with these courses. An even greater problem is that the products of the university, our students, are expected to go out into society as educated individuals. These concrete thinkers can only be described as intellectually crippled.

This brief introduction to the problem is not intended to be exhaustive. The reader is referred to other references if further details are needed.[1]

Enter the Computer

It is the thesis of this paper that interactive student-computer dialogs, where the students interact in a one-to-one or small group basis with computer programs prepared by good teachers, can be an extremely important approach to solving the problems just raised. In this section I will elucidate the reasons why the computer is an appropriate tool.

First, it should be noted that extremely few current university teachers, or teachers at any level for that matter, are aware of the problems suggested. Thus, it is unlikely that our teachers are going to be able to react without either special material or without additional training of some type. Providing more help to students in thinking cannot be expected to come primarily from the existing teaching community unless a sizable effort is made.

Next it is worth noting that there are individuals in various parts of the country who are reporting success in aiding students in making this important progression. Although their effects are expanded by talks at national meetings, at the moment these teachers have mostly local impact. These effects have also been extended by the development, with the cooperation of the American Association of Physics Teachers, of a transportable workshop for instructors. But primarily the experts have only a local effect, not a national one.

There is increasing evidence that the computer can be a very effective learning tool. Indeed, particularly in the sciences, this realization is gaining such common acceptance at the present time that little more need be said. In terms of work in particular areas, perhaps the best sources of information are the various proceedings of the Conference on Computers in the Undergraduate Curriculum for the last ten years. Another interesting source of information are the articles in Change magazine on innovative teaching projects in the United States prepared with a FIPSE grant. While this particular project is not directly concerned with computers, many of the courses described in the Change articles do involve computers.

It is not hard to see why the computer is turning out to be a very effective learning medium. The computer can make the learning experience an active experience for each individual, an experience where the student is constantly participating in the learning process. The situation resembles that frequently seen in a Socratic dialog, where the computer asks questions and the student responds in everyday language. This is to be contrasted with the usual lecture environment where the student is usually in a passive or receptive mode.

The computer also provides an individualized and personalized learning medium not possible in many mass instructional environments such as lectures. The program pays careful attention to what the student is typing and behaves differently for different students. We are not claiming that all existing material is fully personalized. Much of it is not. But enough material exists to show that such individualized interactive capability is possible and practical.

Another important factor in considering computers is the rapidly decreasing cost. Inexpensive, stand-alone, home systems are already available, and costs will continue to decline. The materials described in this report were developed in a timesharing system, so do not represent the latest modern technology. But they could be available on much cheaper, contemporary, home-type computers.

The Dialogs

The six dialogs of interest were developed by the authors in two separate Christmas sessions at the University of California, Irvine. Each occupies approximately an hour's time for the average student, but the deviation is great. The dialogs are as follows:

TERRA — A dialog for discussing the sky as seen from the earth.

LUNA — A study of the phases of the moon, with emphasis on the notion of scientific models and the deductions from them.

HEAT — An introduction to the concept of heat through a series of simple everyday concrete experiences.

MFIELD — An introduction to magnetic fields, treating the field of a long, straight wire, again staying close to the observable phenomena.

MFIELD2 — An extension of the previous dialog to consider the force between current-carrying wires.

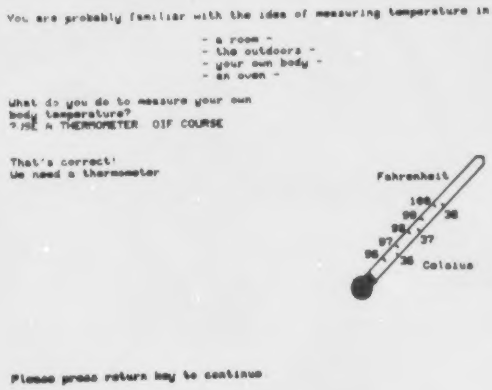
MFIELD3 — Again an extension of the magnetic field ideas, this time to magnetic induction.

LUNA and TERRA have already been described in a previous paper.[2] Space does not permit any full description of the four remaining programs, so we present just one of these, hoping to show the flavor of some of the others. The dialog we will consider is HEAT.

HEAT

The HEAT dialog follows the principles already suggested, introducing abstract ideas, heat transfer and heat, through a series of very concrete experiences. The experiences in this case could all be described as simple kitchen physics. Only thermometers are used as a measuring device. Hot water, cold water, ice cubes, thermos bottles, burners, all enter the discussion. It is the everyday experiences that students have already had with such phenomena that is the principal source of background information for the dialog. Thus, it is expected that students will know that a pot of hot water left on a table for a long period of time will cool down.

The first situation described is simply that of taking one's own temperature. The frame that introduces this looks as follows:

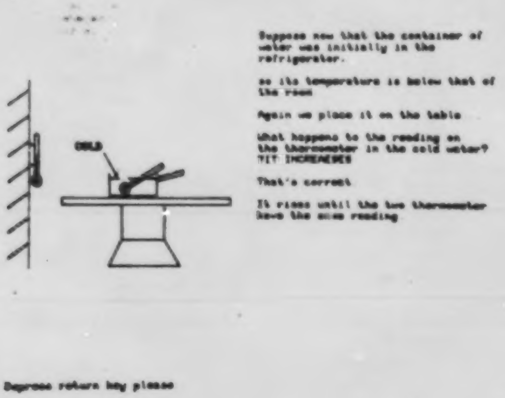


```
You are probably familiar with the idea of measuring temperature in

                    - a room -
                    - the outdoors -
                    - your own body -
                    - an oven -

What do you do to measure your own
body temperature?
?USE A THERMOMETER  OIF COURSE

That's correct!
We need a thermometer



Please press return key to continue
```

Even ten-year olds have no problem answering this question, at least in our society where taking one's temperature seems to be such a critical part of the growing up process! The student is then queried about some of the details of the operation; in any approach to an operational definition it is desirable to get students thinking immediately about the full situation, a measuring device, in this case the thermometer, and a stated procedure for the use of that device.

The next stage involves a pan of hot water sitting on a table and simply cooling off, again an everyday phenomenon that presumably most students will be quite familiar with. We are working even at this early stage toward the notion of equilibrium, getting the student to realize and say that the process will continue until the water reaches room temperature. Again thermometers as measuring devices complement the verbal statement of what happens.

The next sequence repeats the same kind of thing but with a pan of cold water.



Suppose now that the container of water was initially in the refrigerator.

so its temperature is below that of the room

Again we place it on the table

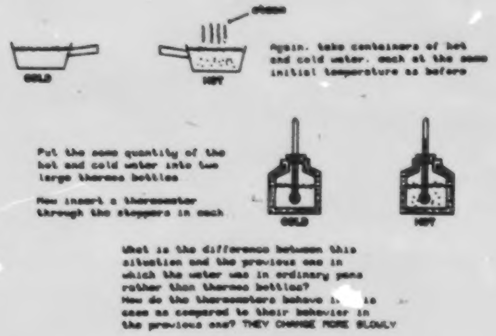
What happens to the reading on the thermometer in the cold water? IT INCREASES

That's correct

It rises until the two thermometer have the same reading

Depress return key please

Now we introduce the notion that the cooling rate may be influenced by various factors by putting water on the table and in thermos bottles, both hot and cold.



Again, take containers of hot and cold water, each at the same initial temperature as before

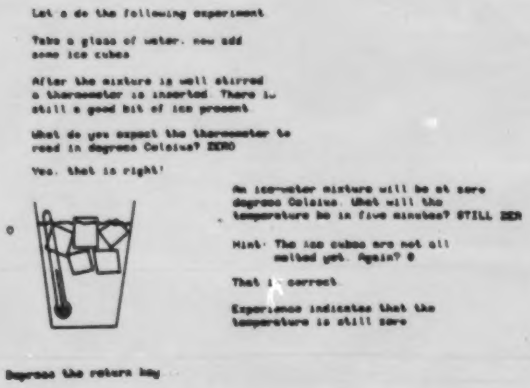
Put the same quantity of the hot and cold water into two large thermos bottles

Now insert a thermometer through the stoppers in each

What is the difference between this situation and the previous one in which the water was in ordinary pans rather than thermos bottles?
How do the thermometers behave in this case as compared to their behavior in the previous one? THEY CHANGE MORE SLOWLY

Again our experience is that the thermos container is a common device in our society. Note that in each case we are drawing on the student's past experiences. If, however, a program such as this were developed for a less affluent society where thermos containers were not commonly used, then we would probably need to follow a different procedure. It is worth pointing out, therefore, that to some extent material of this kind does have obvious cultural dependences, and so cannot assumed to be universal.

The next stage is that of a glass of water with ice cubes sitting in the water. Some time goes into getting the student to explain what happens in this situation, first verbally.



Let's do the following experiment

Take a glass of water, now add some ice cubes

After the mixture is well stirred a thermometer is inserted. There is still a good bit of ice present

What do you expect the thermometer to read in degrees Celsius? ZERO

Yes, that is right!

An ice-water mixture will be at zero degrees Celsius. What will the temperature be in five minutes? STILL ZER

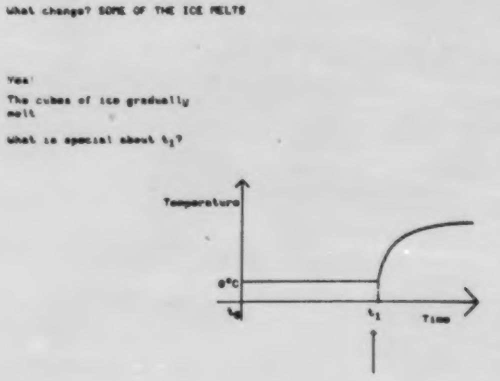
Hint: The ice cubes are not all melted yet. Again? 0

That is correct

Experience indicates that the temperature is still zero

Depress the return key

Then we use graphical aids.

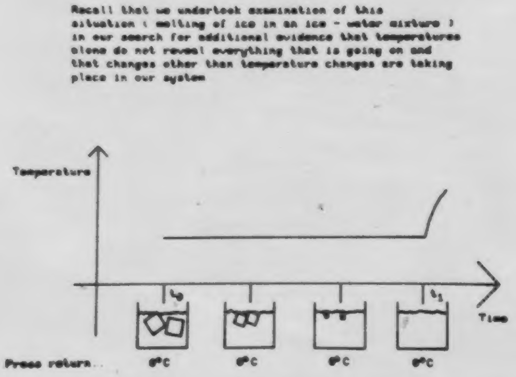


What change? SOME OF THE ICE MELTS

Yes!

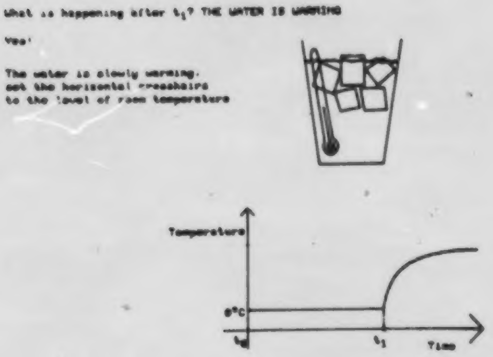
The cubes of ice gradually melt

What is special about t_1?



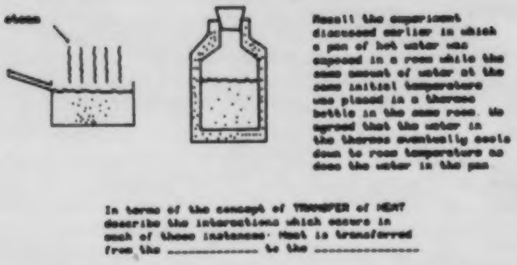
Recall that we undertook examination of this situation (melting of ice in an ice - water mixture) in our search for additional evidence that temperatures alone do not reveal everything that is going on and that changes other than temperature changes are taking place in our system

Press return...


What is happening after t_1? THE WATER IS WARMING

Yes!

The water is slowly warming.
Set the horizontal crosshairs
to the level of room temperature



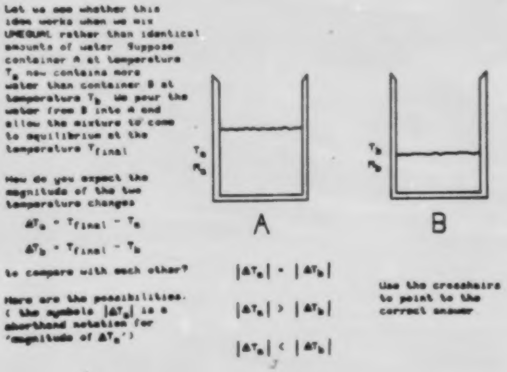
Recall the experiment
discussed earlier in which
a pan of hot water was
exposed in a room while the
same amount of water at the
same initial temperature
was placed in a thermos
bottle in the same room. We
agreed that the water in
the thermos eventually cools
down to room temperature as
does the water in the pan

In terms of the concept of TRANSFER of HEAT
describe the interactions which occur in
each of these instances: Heat is transferred
from the ____________ to the ______________

Again the graphic display might have to be modified for students at a stage where graphic information is not understandable.

It is at approximately this stage that we begin to suggest verbally what we have been driving at. Up to this point the student is seeing one simple experiment after another and talking about what is happening. Now, however, we have sufficient evidence gathered in this slow, careful way to begin to try to draw conclusions.

After the initial verbal ideas the dialog goes through yet more experiments, now testing how this type of explanation works with other phenomena besides those that we have already seen. Thus, the mode is something of a classical, scientific approach, developing ideas and then seeing if the ideas work in new experimental situations. Eventually some mathematical notation appears.


Let us see whether this
idea works when we mix
UNEQUAL rather than identical
amounts of water. Suppose
container A at temperature
T_a now contains more
water than container B at
temperature T_b. We pour the
water from B into A and
allow the mixture to come
to equilibrium at the
temperature T_{final}

How do you expect the
magnitude of the two
temperature changes

$\Delta T_a = T_{final} - T_a$

$\Delta T_b = T_{final} - T_b$

to compare with each other?

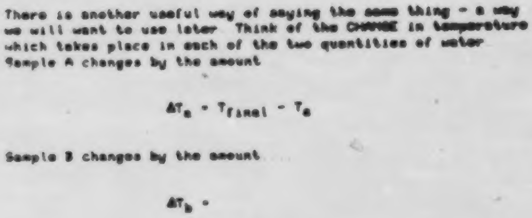
Here are the possibilities.
(the symbols $|\Delta T_a|$ is a
shorthand notation for
'magnitude of ΔT_a')

$|\Delta T_a| = |\Delta T_b|$

$|\Delta T_a| > |\Delta T_b|$

$|\Delta T_a| < |\Delta T_b|$

Use the crosshairs
to point to the
correct answer


There is another useful way of saying the same thing - a way
we will want to use later. Think of the CHANGE in temperature
which takes place in each of the two quantities of water
Sample A changes by the amount

$\Delta T_a = T_{final} - T_a$

Sample B changes by the amount

$\Delta T_b =$

We are only showing a small fragment of HEAT. For a full version the reader should gain access to and run the program.

Final Comments

The four programs, and many others, are accessible via phone lines to those who would like to view them and who either have or can acquire Tektronix 4010 series graphic displays. The project also encourages visitors and has considerable additional literature available concerning other activities.

These few programs suggest some very interesting directions for the future. The problem raised, that of students who have not reached their full intellectual development, is a serious one in our society. More programs of this kind, available in a wide variety of environments, could have a major, national impact on the intellectual life of the country. We would like, for example, to develop further material specifically for public libraries and science museums. They could also be widely used in such other environments as shopping centers and airports. Computer dialogs have sufficient appeal in their own right to compete well with the other types of activities in those locations.

Often too the process of learning the

subject matter content could be an important secondary function. Thus, we can conceive of doing the same type of programs in such areas as energy conservation, not only helping people to become formal reasoners in a full sense but also helping them to understand the notions of energy and the importance of the reasonable use of energy.

The three magnetic field dialogs were the product of the two authors. HEAT involved a third member of the University of California, Irvine group, Joseph Marasco, codirector of the Physics Computer Development Project at Irvine. The student programmer on the magnetic field dialogs was Martin Katz, and the student programmer on the HEAT dialog was Tod Hopson. In both cases the programming work was managed by the senior programmer of the Physics Computer Development Project, Estelle Warner.

The programming aspect of the dialogs, done by student programmers, took approximately three man-weeks for each of the dialogs. Since the students did not work a 40-hour week, the work was spread over a longer period of time. This figure includes time for the extensive revision which characterizes the earlier development of such products.

Much of the interactive design of the material, the composition of both graphics and text on the screen, was done using an extensive screen design capability developed at the University of California, Irvine project over the last several years. Almost all of the above individuals have played major roles in the design of that product. The programmer for the entire history of the development of the product was Mark Geisert.

References

1. Arons, A. B., "Cultivating the Capacity for Formal Reasoning: Objectives and Procedures in an Introductory Physical Science Course," Am. J. Phys. 44, 1976, pp. 834-838.
   Arons, A. B., "Some Thoughts on Reasoning Capacities Implicitly Expected of College Students," Proceedings of the Conference on Cognitive Process Instruction, University of Massachusetts, Amherst, June 1978.
2. Arons, A. B. and Bork, A., "Two New Graphic Computer Dialogs for Teachers," Am. J. Phys. 43, No. 11, November 1975.

DEVELOPMENT OF A RESOURCE CENTER
FOR
INTRODUCTORY COMPUTER SCIENCE

by

J.A.N. Lee
Virginia Tech

ABSTRACT

This paper reviews the experiences of developing a Computer Sciences Resource Center through the joint efforts of faculty and students, at minimal cost to the institution, yet with maximal opportunity for learning experiences by students even in the initial offerings of the courses. Although especially directed at the so-called Computer Appreciation courses, the resources developed through this program are applicable to other computer science courses and to other presentations across the college campus.

INTRODUCTION

The opportunity to develop meaningful learning experiences in the field of computer science has often been lead astray by the concept that nothing meaningful can be presented to participants unless there is a large dose of computer interaction. Whether this is accomplished by means of batch operations through a closed shop operation or through an interactive laboratory, the cost per student is often sufficiently high, that its use is restricted to only those students who have a need to know.

The project which is described here bases its syllabi and supporting materials on the concept that there is much more to computer science than just programming. In particular, in the presentation of Computer Appreciation to non-majors, every advantage has been taken of their native interests in order to achieve two intrinsic goals. Presentating material that was highly pertinent to the major interests of the students in the courses was considered a necessary goal in order to sustain their interest and further to include them (and consequently their faculty) in further course development. Secondly, the skills that these non-majors possessed were to be put to work in assisting in developing resources for succeeding courses.

COURSE CONTENTS

The Department of computer science at Virginia Polytechnic Institute and State University (Virginia Tech) is responsible for the presentation of beginning courses in computer science for the whole institution. This responsibility is based on the notion that there needs to be standardization of presentation of materials at this level. In particular, the department has emphasized current techniques and technologies in these courses prior to the time when other non-specialist departments have become aware of the trends. Thus the inclusion of disciplined programming techniques in introductory programming courses has been easily accomplished.

Notwithstanding the centralization of instruction at the introductory level, the department and college have encouraged the development of applications courses by other departments in the University. Thus the School of Architecture has a broader offering in computer graphics than the Department of Computer Science, and the departments of Chemistry and Chemical Engineering have more students registered in courses on TTL circuits, minicomputer interfacing, and microprocessors than the combined departments of Computer Science and Electrical Engineering.

The majority of students registered in introductory computer science courses are there simply because of departmental and college graduation requirements. While we recognize that the reason for this requirement in science and business major curriculum is to provide a base for the use of computers in later applications courses, the presentation of material to non-science, non-technical majors is not so cut and dried.

It is interesting to contrast the learning objectives of these two styles of curricula in introductory courses:

For introductory programming courses:

> The purpose of an introductory programming course is to acquaint the students with the principles, methods, and techniques of converting problem solutions into computer programs.
>
> The major desired outcome of this course is to provide a base for building an ability to apply the computer to his own major field of interest.
>
> As a result of this course a student should be knowledgeable about the techniques of programming in one particular language and be able to recognize the applicability and limitations of the computer.

For computer appreciation courses:

The purpose of this course is acquaint students with the basic concepts of computers, computing, and computation, through studies of the historical development of the science, the fundamentals of computer hardware and software design, and applications of the computer technology to his major field of interest.

The major outcome of this course is to broaden the student's knowledge of the technology to which he is subjected day by day and to understand the applicability and limitations of computer technology.

As a result of this learning experience, the student should be able to understand the applications of the computer to his major field of interest, to be able to judge the usefulness of attempting to use the computer in various situations to which he may be exposed, and to possess the necessary vocabulary to interact with professional computer personnel.

It is to students in these latter courses that we turn our attention now.

In order to achieve the learning objectives, it was determined that the course(s) needed to cover three essential aspects of the subject; the historical development of the subject, the current technology, and the applications of that technology. The content of these courses conforms closely to the knowledge base being considered by the Elementary and Secondary Schools Subcommittee of the Education Committee of ACM [1,2]. Included in the historical aspects are several topics which can be developed extensively. In fact, it has been our experience that that there is so much material here that the tendency to revel in the intracasies of the history must be subjugated to the needs of the students. However, like the other areas of study, the complete presentation of the material is impossible and thus it is necessary to provide opportunities for the students to further investigate specific topics themselves. Included in the historical topics are three elements which have been found to be of specific interest to students in other disciplines. These are studies of the progression of development of certain concepts (such as number systems, codes and cyphers, the elementary computer, etc.); studies of the people involved (such as Babbage, Napier, or even more recent figures as Grace Hopper, Herman Goldstine, and others); and studies of the construction of specific devices (such as Jacquard's loom, Hollerith's card processors, etc.) and their effects on the problem solving community.

To know about current technology students need to understand the basics of both hardware and software. It is the intensity of presentation of this aspect of the subject area that distinguishes the various introductory computer science courses. For the student in the humanities, the material presented covers logic and circuitry sufficiently to develop (say) a full adder and thus to realize the simplicity of the construction of a machine, sufficient exposition on the construction of computer peripherals and their media (such as magnetic tapes and disks) sufficient to provide an understanding of the options available to the user of a system, and knowledge of programming and languages sufficient to realize that the development of programs is neither a mystic art nor a faultless activity.

The applications presentation is divided into two specific subject areas: the application of the computer to the student's major field of interest (or vice versa) and the general application of computers to daily life. The former topic is covered by independent student projects while the latter, including such topics as social implications of computer usage and "way out" computer applications such as artificial intelligence and robotics, is covered in classroom activities.

METHODS OF DEVELOPMENT

While many university educators have decried the ability of the average student to express himself clearly in his native language, it has been our experience that the average computer science student is below the average with respect to this measure. The students from other disciplines such as journalism, interior design, or education have skills which can be tapped for the benefit of all students. Thus our Computer Appreciation course (more aptly titled Computer Concepts) has become the foundation of an ongoing resource development project within the Department of Computer Science.

This particular course has been taught on both the semester system (at the University of Massachusetts at Amherst as COINS 101, Introduction to Computer Science) and during a quarter (at Virginia Polytechnic Insitute and State University as CS 1020, Computer Concepts). Obviously the difference in time periods made a significant difference to the amount of material covered, but the enthusiasm of the students to assisting in the development of course materials as part of the assignments has not diminished.

Initially, (in 1971, when the course was first offered at the University of Massachuetts at Amherst) the projects chosen for student preparation were totally the brain-child of the faculty member and left no room for student options. Each of the assignments was specific, though the mode of presentation of the student's efforts was not specified. Since that time, student assignments have been chosen on the basis of University, departmental, and course needs, and have been widened to provide the student with as many options as possible. The need for such options was necessitated (in my opinion) by the need to match course requirements with students' skills. That is, every opportunity should exist for the students to express themselves in the medium which they prefer.

The results of this experiment have been extremely satisfactory. Not only have the students benefited from the opportunity to show their knowledge of the subject in terms of their own skills, but also the whole University community has benefited. For example, a student of agriculture choose to research the application of the computer to the dairy industry and produced a 17-minute tape and slide presentation on genetic mating services. This presentation is now in use in the Department of Computer Science for succeeding generations of the course in which it was developed and also in several other introductory courses. At the same time, the Department of Dairy Science possesses a pedagogical tool for introducing the computer to its students, whilst showing off a faculty member's research topic to other interested parties. This latter group includes presentations at University career days, prospective student groups, high school visitations, and accreditation committees. Similarly, a student's project on the applications of computers to air traffic control, chosen by a student on the basis of her father's occupation, lead to a closer relationship between the Department of Computer Science and an ongoing radar-related research project, and aroused several other students' interest in aviation related projects.

To identify certain University needs, close cooperation is maintained between the course director and two other campus activities; the Computing Center (through User Services and the Publications Division) and the Learning Resources Center. The latter group have themselves been essential to the success of this project by providing consultation and services with respect to the production of instructional aids. As a source of project topics, both cooperating agencies are able to provide information on requests, initially directed to them, which can be satisfied by student projects. For example, a recurring request of the Computing Center was to provide instruction or materials on the use of key punches. By simply assigning the development of poster-sized instructions on the use of a key punch as one of the optional projects in the course, there now exist displays in each key punch room on the campus. This project helped students in the course to understand the use and operation of a key punch for later programming activities.

The AV needs of the course are often met by the students themselves by proposing alternative assignments. For example, for the first year of the course, the explanation of basic computer organization and the presentation of a simple machine language was accomplished in terms of the CARDIAC computer [3]. This computer was chosen not only for its simplicity and generality, but also because of the low cost of the cardboard computers themselves. It was a student who proposed, as his final project in the course, to write a CARDIAC simulator in the BASIC language so that students could actually run CARDIAC programs. In a later course (at a higher level) a symbolic version of CARDIAC called ASCARD (ASsembler for CARDiac) was developed by students who had earlier participated in the Computer Concepts course. This processor is now in use in the introductory courses as a simple example of an assembly language.

COST EFFECTIVENESS

The cost of operating this course and hence the cost of developing the resource center depends on two factors: how much use is to be made of commercially available aids and how much is to be developed locally. The experience of this author has been that starting with a shoe string budget, not being overly concerned with the professional quality of the aids initially produced (though obviously there was always deep concern with respect to the learning experiences of the students in developing the materials and the accuracy of their presentations), lead to a snowballing effect. Otherer agencies besides the Department of Computer Science became interested and involved, so that both funding and technical support became more readily available. Both at the University of Massachusetts and at Virginia Polytechnic Institute, college agencies concerned with the improvement of teaching was faculty eager to assist, and both institutions supported continuing work by providing Teaching/Learning Grants. Thus the cost of curriculum materials is well within the budget of the smallest department and is infinitessimal when contrasted with the cost of operating a quarter-long programming course.

CONCLUSIONS

The cost of providing hands on computer experience for non-science (and non-engineering) majors has been so high that there has been a tendency to provide a totally different kind of course for them. Often this has been in the form of a Social Implications course, where the cost of presentation has been so minimized that the student receives little direct contact with the computer. Whilst such courses have their place, it is the premise here that an understanding of computer technology is equally important, and that such an understanding can be achieved without incurring high costs. In fact, the cost of procuring materials is minimized to the acquisition of the raw materials themselves, such as poster boards, film, cassette tapes and artist's materials. The enthusiasm of students and the skills which they have developed in other fields, together with the broad base of contacts they have outside of the field of computer science, provide a basis for the establishing and continuing development of a Computer Sciences Resource Center at any institution.

One measure of our own success may be the freedom with which faculty utilize the available materials in other courses, and then assist in the development of additional materials. Once the center was been established it was surprising how much material in the way of computer parts (old disks, magnetic tape heads, cores etc.) was accumulated from the faculty. Faculty have also contributed their help to construct devices to aid in the teaching of various concepts. The major drawback to this project has been the amount of time necessary to produce the first few aids and to set the example for succeeding developments. However, where a department has an independent study course, students in the course can research the basic topics and produce the core set of aids. Thereafter, the course itself will self generate its own materials.

One possible side effect of this program of development which cannot be overlooked is the improvement of town/gown relationships. This benefit was particularly evident when a student assignment investigated the application of the computer to a local industry. It had been anticipated that students would descend upon the local manufacturing industry, and some advanced preparation with the computing center directors and managers had prepared for the expected student enquiries. The flood of enquiries did not materialize; instead the students had made their investigations much closer to home — in the local bookstore which had recently installed a point of sales operation; in the discount store that used computerized clothing tags; in the catalog store which used a teletype to send in their daily orders; and in the tire company who used hand printed invoices which were later processed by computer. The ingenuity of the student, as usual, was overwhelming.

Finally, to ensure that the success of the course was not merely dependent on the enthusiasm of a single faculty member, the same course has now be administered (one can hardly say taught) by two other faculty, each with comparable results.

ACKNOWLEDGEMENTS

The author wishes to acknowledge the assistance given during this project by Mike Melnik of the University of Massachusetts, Drs. Braden and Head of the Learning Resources Center, Instructional Development Division, Virginia Tech, and for the inspiration to develop a center from Dr. Ruth Hoffman, Mathematics Laboratory, University of Denver. Acknowledgement is also due to Nina Thayer of Virginia Tech, for her work on developing additional materials for the center and for the photographs contained herein.

REFERENCES

[1] Hagelbarger, D., and Fingerman, S., An Instructional Manual for CARDIAC, Bell System Educational Aid, Bell Telephone Laboratories, 1968. (Now distributed by AMF Industries, Arlington, VA.)
[2] Johnson D., et al, "Computer Literacy - What is it?", Minnesota Computing Consortium, 1978, to be published.
[3] Lee, J.A.N., "The Literacy Test 1978, A Preliminary Report", Virginia Polytechnic Institute, 1978, to be published.

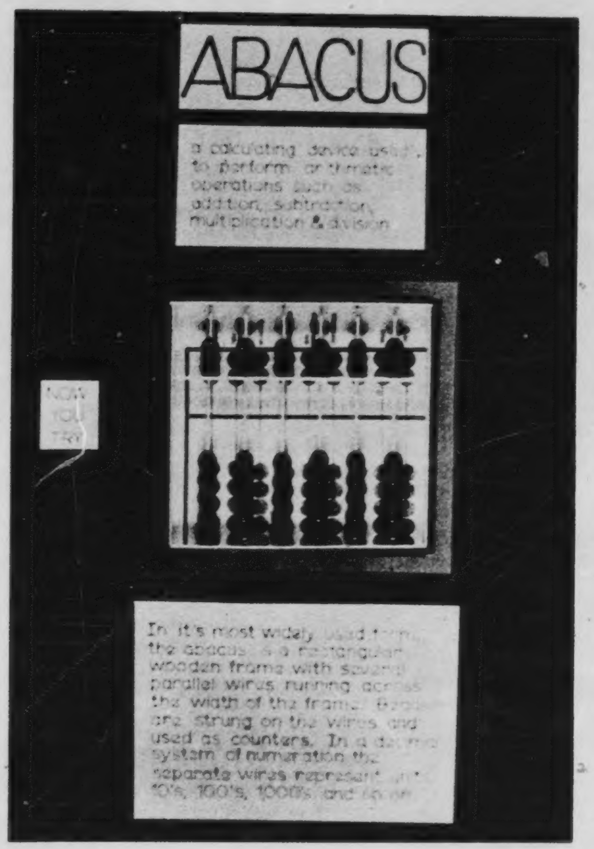
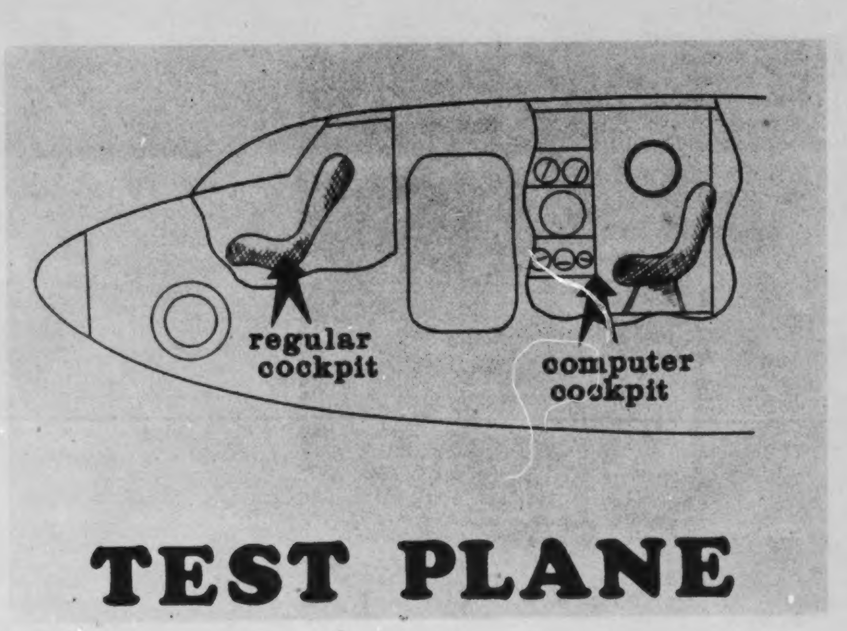

ABACUS
a calculating device used to perform arithmetic operations such as addition, subtraction, multiplication & division
NOW YOU TRY
In it's most widely used form, the abacus is a rectangular wooden frame with several parallel wires running across the width of the frame. Beads are strung on the wires and used as counters. In a decimal system of numeration the separate wires represent 10's, 100's, 1000's and so on

Figure 1

Part of a three-element display on the abacus



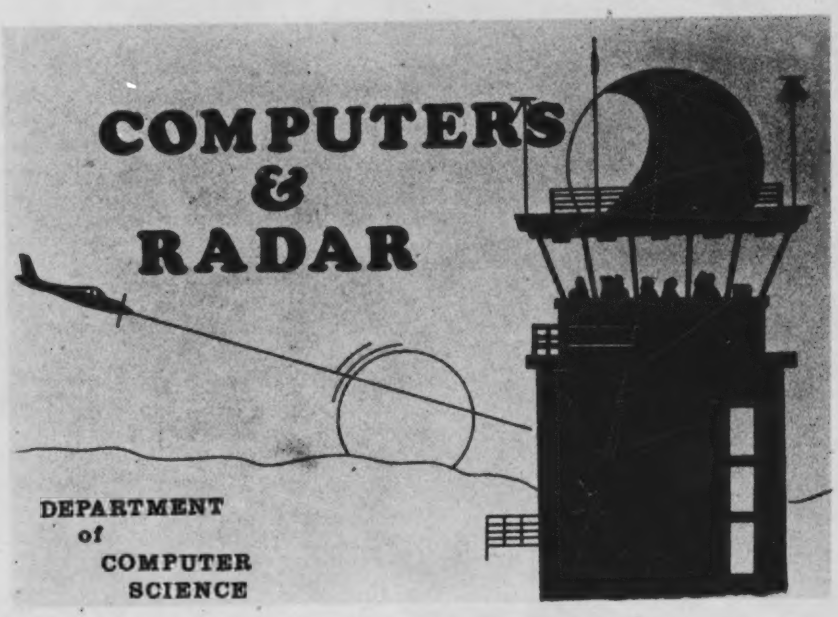
regular cockpit

computer cockpit

TEST PLANE

Figure 2a

Slide and tape presentation on the Application of Computers to Aircraft Control



COMPUTERS & RADAR

DEPARTMENT of COMPUTER SCIENCE

Figure 2b

A slide from the Computers and Radar presentation

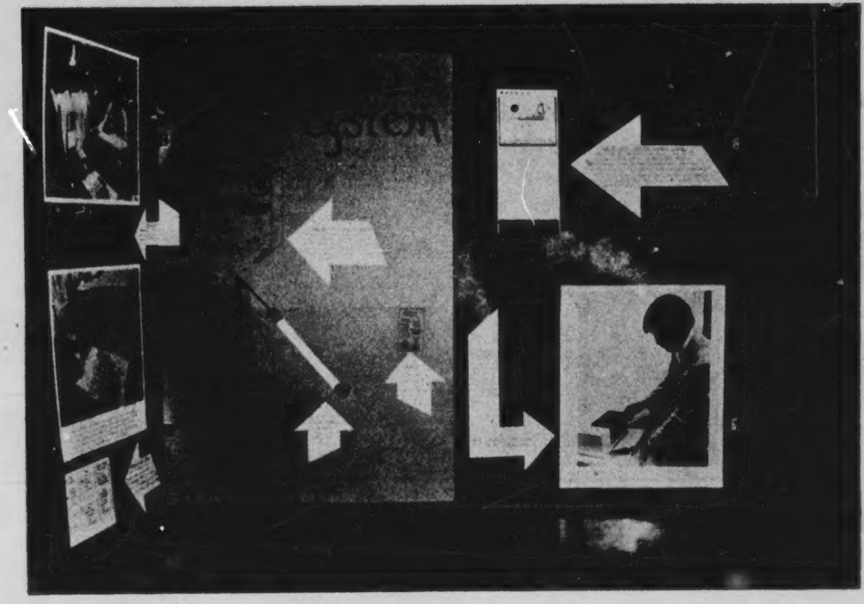


Figure 3

A visual poster display on Point of Sales Applications

Figure 4

A flip chart of Supermarket Scanning Devices

WHAT PROGRAMMING LANGUAGE, IF ANY, FOR BEGINNERS?

Herman D. Hughes
Michigan State University

ABSTRACT

Introductory courses, originally developed around programming languages, have been slow to move away from the emphasis on language syntax. What needs to be emphasized in these courses is a systematic, well-disciplined approach to problem-solving. This paper presents a brief discussion on typical programming languages used by beginning programmers and attempts to establish a set of desirable features for any programming language. In an effort to address the problem of deciding on a programming language for beginning programmers, a language is presented that is designed to enable users to concentrate on problem-solving instead of syntactical details. An example illustrates the use of the language. Finally, some results are discussed in regard to using this language in a beginning programming course at MSU.

I. INTRODUCTION

Many introductory computer science courses at Michigan State University and elsewhere require students to obtain the correct solution to several programming exercises. Unfortunately, the efficiency of the algorithms and the structural procedures used for determining these solutions are generally not emphasized by either the students or the instructors. Consequently, students may acquire undesirable programming habits beginning with their first programming course; the structures of their programs do not clearly reflect the structures of the problems they represent and the patterns of analysis that the students have followed. Unless students are sensitized to good problem-solving techniques at the beginning level, poor programming practices seem to develop.

In order for any programmer to be sufficiently motivated to attack problems in a systematic manner, a sense of payoffs must exist at each phase of the problem-solving process: 1) the problem analysis phase, 2) the algorithm development phase, and 3) the computer implementation phase. Perhaps equal weight should be given to each of these steps when students' programs are evaluated.

A major problem in introductory computer science courses is that the programming languages taught are those used by professional programmers; such languages (i.e., FORTRAN, BASIC, COBOL, PL/I, etc.) either overwhelm a high percentage of beginning programmers because of their size and complexity, or have structures which are not conducive to problem-solving. As a result, emphasis is placed on language syntax instead of problem-solving concepts. For example, in a typical FORTRAN course, two weeks or more may be spent on format statements for I/O; this material, while necessary for detailed work in FORTRAN programming, has no relevance to the logical solution of problems.

Efforts have been made to resolve the problem of what programming language, if any, should be taught in beginning programming courses. One approach is to use a high-level pseudo language (i.e., ALGOL-like notation) for describing algorithms, and then use a high-level language such as FORTRAN for the computer implementation of the algorithms. Universities using this approach should be commended for their boldness, but I fail to see how this two-stage process is an improvement of the status quo. Another approach to language selection for beginners is to use a high-level language for describing algorithms, and then use this high-level language for the computer implementation of the algorithms. This approach appears to be attractive, provided that the high-level language is simple and possesses structures which facilitate problem-solving concepts. The debate on the appropriate language for beginning programmers will probably not be settled in the forseeable future because of: 1) equipment and software considerations, and 2) the massive number of computer science educators who declare themselves as experts on this matter, and therefore, consider their opinions as final. What is encouraging, however, is that computer scientists generally agree that our traditional approach to teaching computer programming needs a careful appraisal. Such an appraisal is sure to lead to recommending innovative strategies for introducing computer programming. One proposed strategy forms the basis for this document; this strategy should shed some light on the argument what programming language, if any.

This paper presents the results of a research project designed to develop a simple, high-level programming language in which users of varying technical backgrounds can conveniently describe algorithms and simultaneously focus on problem-solving concepts. The language is called A Structured Programming Language (ASPL).

II. SOME DESIRABLE FEATURES OF A PROGRAMMING LANGUAGE

In view of the significance of structured programming, programming languages should contain constructs that encourage the clearest possible statement of the program's goals. Furthermore, language should be simple, powerful, and natural. They should aid programmers in both writing and reading programs. Programming languages must also provide convenient mechanisms for modifying programs and verifying their correctness. The two most frequently used high-level languages, FORTRAN and COBOL, do not support all the attributes for developing well-structured programs. Consequently, several translators [1] have been built for translating structured FORTRAN and structured COBOL into standard FORTRAN and COBOL. Certainly, these translators are valuable tools, but the programmer can be baffled when attempting to follow a program's logic from the code written for the translator to the code produced by the translator. It is important that the transformation of a programmer's logic from the design phase to the coding of the program be a straightforward process. A good programming language must have structures that appeal to the users' thinking processes at the highest level of the design phase. Since there appears to be a relationship between the features of a programming language and the thinking habits of programmers using that language, the language should provide a set of features that stimulate problem-solving techniques that are transferable to other languages as well as applicable to solving problems in general.

Two conventional programming languages, PL/I and PASCAL, have the constructs for structured programming. PL/I is perhaps the most powerful, high-level general purpose language on the market. It is an extremely large language, and thus, requires considerable time for programmers to learn and use efficiently. On the other hand, PASCAL is a much smaller language than PL/I, and it is also less powerful in terms of character manipulation and file handling facilities. Because of the size and complexity of both PL/I and PASCAL, programmers (especially beginners) can easily be overwhelmed and forced to concentrate on syntactical details instead of problem-solving concepts. SP/K, a reduced version of PL/I was designed for the purpose of providing users with a small language, but it, too, is large and complex, and does not have enough properties to encourage good problem-solving techniques.

The author does not claim to have a precise prescription for the set of features to be included in a language, but does believe that adding new features to existing programming languages (i.e., FORTRAN '77 and COBOL '77) is the wrong approach because the results are larger and more complicated languages with meager gains in power for problem-solving methodology.

III. ASPL AND OTHER LANGUAGES

This author is aware of a few universities that are experimenting with the implementation of a small, high-level language for teaching beginning programmers, but has not seen any published materials regarding either the success of these efforts or any details on the nature of the languages.

ASPL differs from other conventional, high-level languages (i.e., FORTRAN, PASCAL, COBOL, PL/I) in several ways: 1) it is a much smaller language, 2) it contains only the minimal basic structures, 3) it does not allow statement labels and GO TO statements, 4) only top-down programming is possible, 5) it requires the user to specify type on the variables only if they are to be declared as character, otherwise all variables and constants are treated as reals, 6) only simple I/O is supported, and 7) it runs as an interpreter.

Since decision tables form an important feature of ASPL, it is only fitting to mention other uses of these tables for programming purposes. Several decision table processors are available and some of these were developed as early as the mid-1950s [9]. DETAB-X is a decision table preprocessor which accepts the decision tables as input and gives a form of COBOL coding as output [10]. FORTAB and CENTAB accept decision tables as input, and output a form of FORTRAN coding [11].

More recently developed decision tables processors have been reported [12], all of which have characteristics similar to the previously developed processors. Since these processors were developed by private industries and government agencies, they are designed for specialized applications such as complex file maintenance problems [11]. Furthermore, the processors accept decision tables as input and output a form of programming language such as FORTRAN, COBOL, PL/I, etc. [11]. Hence, a three stage process: 1) decision table construction, 2) code generation, and 3) compilation and execution, is necessary for the user to obtain the final output from his program. This three stage procedure used by existing processors, coupled with the design specializations, make the available processors less appealing in a college or university environment. Also, programmers may encounter grave difficulties when attempting to debug their programs because of the excessive number of programming stages. Therefore, a language which is interpretive and also permits decision tables as one of its constructs, seems to have great promise.

Since ASPL runs in an interpretative mode, the cost of running programs is slightly higher (about 10%) than that for running the FORTRAN equivalent programs. This extra cost can be justified on the basis that good run-time diagnostics (e.g., subscript range checking) are available to users, and that on the average fewer runs are required for the successful completion of programs.

IV. STRUCTURE OF AN ASPL PROGRAM

In ASPL, there are three types of structures: 1) PROGRAM, 2) BLOCKS, and 3) SUBBLOCKS. These three structures form a hierarchy in which the PROGRAM structure contains BLOCKS and SUBBLOCKS, and BLOCKS logically contain SUBBLOCKS. This hierarchy structure is depicted in Figure 1.

BLOCKS are disjoint structures, and may be thought of as mini-algorithms that make up a main algorithm. They correspond to breaking down an algorithm into large general steps. Each of these steps may have many details to complete before the general steps are done. In fact, these general steps are the intermediate goals that must be achieved before we can accomplish the overall goal of the algorithm.

There is one additional structure to be considered along with PROGRAMS and BLOCKS, and that is the SUBBLOCK. Although the general form of a SUBBLOCK is similar to a BLOCK, its logical relationship to a problem and to other program structures is somewhat different. A SUBBLOCK, like a subgoal, may have to be executed (reached) more than once before solving the problem, and different conditions may cause the SUBBLOCK to be invoked. In other words, this subgoal is a common subgoal.

Figure 1 shows the logical relationship of BLOCKS and SUBBLOCKS to a program. The important thing to notice about the diagram is that for the execution of a program, only BLOCKS are automatically executed in the order of their appearance in the program. SUBBLOCKS, however, are skipped over because the programmer must initiate the execution of a SUBBLOCK by using a PERFORM statement within a BLOCK or SUBBLOCK. The numbers opposite the arrows indicate the sequence of flow (i.e., 1→2→3→4→5→6→7) for the program structure given in Figure 1. For example, execution will begin at the statement BLOCK 1, and continue through the statement PERFORM 1. Then, execution will proceed at the statement SUBBLOCK 1, and continue through the statement PERFORM 2, etc.

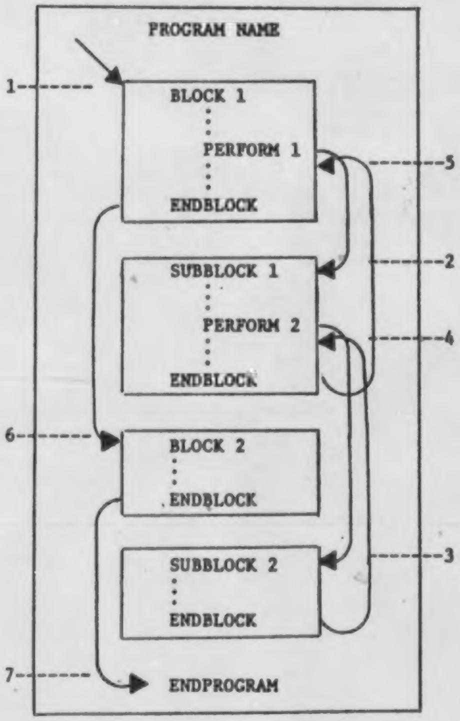


Figure 1.
Program Structure with
BLOCKS and SUBBLOCKS

V. PROGRAMMING STYLE

The style used by programmers to structure their program is very important. Programming languages should be engineered in a way that a systematic indentation of users' programs is an inherent feature of the languages. Systematic indentation of programs makes them easy to read and can assist in locating logical errors since this hierarchical arrangement allows a quick grasp of the structure of the code. Also, the indentation of code segments facilitates their verification. Other attributes of programming style are: 1) restrictions on statement lengths and 2) limitations on module sizes.

In ASPL, programming style is enhanced by the availability of the following features:

1) The automatic indentation of statements to emphasize the structures involved and the nesting of these structures. For example, the IF-THEN-ELSE and ENDIF are placed on separate lines with the same margin. The code following the IF-THEN-ELSE is also indented with the same margin. Other constructs placed on the same margin include: DOWHILE and ENDO, BLOCK and ENDBLOCK, and SUBBLOCK and ENDBLOCK. Decision tables are alligned in columns to facilitate easy reading.

2) The restriction of relatively long, complicated statements within a program.

For example, the statement:

```
Y=(13*X+6-5*Y)-6*X+4*Y**3/(s*X+3*Y)+X-
   Y/3-3*(X+Y)
```

is too complex and should be written as two separate statements and then combined by a single statement. In ASPL, long statements are discouraged, since there are no provisions for the continuation of a statement onto the next card or line. NOTE: The maximum number of characters permitted in an ASPL statement is 80.

3) The mechanism which permits the construction of a program so that it is modular. In ASPL modularity can be achieved by the use of BLOCKS and SUBBLOCKS. The programmer is forced to use modular programming techniques in ASPL because it allows only one IF-THEN-ELSE construct per BLOCK/SUBBLOCK. Statements appearing within the BLOCKS/SUBBLOCKS may consist of the basic control structures (i.e., sequential, DOWHILE, IF-THEN-ELSE).

VI. PROBLEM-SOLVING IN ASPL

ASPL is perhaps most powerful for solving problems in which many conditions exist and there are several alternatives for satisfying these conditions. Problems which involve logic for playing games (e.g., checkers, BlackJack, Roulette, Space Wars, etc.) are typical examples. Still another class of problems that lends itself to ASPL solutions is simulation. The following example illustrates the use of ASPL for playing a card game.

BLACKJACK GAME (MODIFIED)

We are going to develop an algorithm for a modified version of the card game of BLACKJACK (or 21). The strategy of this game is to draw cards until your score is closer to 21 than your opponent's score, without either score exceeding 21. For example, assume that the game consists of two players with one player acting as a dealer. Each playing card has its face value, except for cards over 10 (i.e., Jack, Queen, and King) which have the value of 10; an ACE can have the value of either 11 or 1 (depending on whether drawing an Ace makes your score exceed 21 or not). The set of rules to be used for our version of BLACKJACK is given below. The two competitors will be referred to as Player 1 and Player 2 (the dealer).

1) If Player 1 has exactly 21, he wins unless the dealer's score is 21.

2) If both players have scores less than 21, the closest one to 21 wins.

3) If Player 1's score is less than 17, he must draw another card.

4) If Player 1 has drawn 5 cards and his score does not exceed 21, he automatically wins.

5) If the scores for both players are equal or both scores exceed 21, the dealer (Player 2) will be the winner.

For each game, Player 1's score will be determined by drawing cards (i.e., generating random numbers from 2-14) until one of the rules cited above is satisfied. It should be noted that a card has the value of 10 if its face value is either a 10, Jack, Queen, or King. The random numbers 10, 11, 12, and 13 are generated to effect the correct probabilities for getting a 10. The random number 14 is generated to represent an ACE, and this card will either be assigned the value 11 or 1.

Boxes 3 and 4 of Figure 2 form a decision table. The symbols "Y", "N", and "-" denote YES, NO, and DON'T CARE, respectively. Next, we need to determine what variables are needed to refine the general algorithm. Since playing the game, determining the winner, and reporting the results seem to be the natural groupings of the general algorithm, the refinement process will involve an analysis of: 1) boxes 1 and 2, 2) boxes 3 and 4, and 3) box 5. First, let us consider a finer breakdown of boxes 1 and 2. This breakdown is illustrated in Figure 3 by boxes 1-8.

It should be noted that box #6 (PERFORM CARDVALUE) needs to be refined in order to show how the face value of a card is determined. Also, box #7 (PERFORM WINNER ROUTINE) must be refined so that the logic for deciding the winner is clear. Finally, box #8 needs refining in order to show how the champion is selected. The refinement of only one of the above boxes is given (refer to Figure 4). The refining of the other two boxes can be done in a similar manner. For additional details of the logic for these boxes, refer to the code in Figure 5.

Boxes 1-6 determine the total points for each player, and box #7 determines the winner. A refinement of box #7 is given in Figure 4.

It can be seen from Figures 2-5 that the transformation of the program's logic from the design phase to the coding of the program is a straightforward process. The verification of the correctness of the program is a relatively easy task. In particular, the decision tables enhance this effort.

Each BLOCK/SUBBLOCK of the program was designed and tested separately, starting with the top-level BLOCK (i.e., BLOCK 1). Dummy SUBBLOCKS were set up for those SUBBLOCKS in which code did not exist so that the top-level BLOCKS could be run and tested. After finding and correcting the errors in the upper-level routines, the dummy SUBBLOCKS set up for the top-level BLOCKS were filled in, and new dummies were established, if necessary, to keep the partially completed program running. This process continued one BLOCK/SUBBLOCK at a time until the complete program was running correctly.

ASPL provides a comprehensive set of compilation and execution-time diagnostic messages that made the debugging of the BLACKJACK program relatively easy. For example, some of the messages which are produced are:

```
        :
        :
        PLAY1TOL  GE  17;
              ↑
*****    WARNING:  ERROR NO. 36--NON-FATAL:
         VARIABLE NOT DEFINED
        :
        :
        PERFORM CARDVAL
                      ↑
*****    COMPILATION--ERROR NO. 29--FATAL:
         NO (SUB) BLOCK DECLARATION
        :
        :
```

ASPL also provides a dump of all variables and their values when a fatal error is encountered during program execution. Alternatively, the user can specify that certain variables are dumped from selected BLOCKS/SUBBLOCKS. Finally, a trace to indicate which rule of a decision table was activated for a BLOCK/SUBBLOCK is also available.

VII. EXPERIMENTAL PROJECT WITH ASPL

During the 1978 spring term, a group of beginning programmers were presented ASPL instead of FORTRAN as the primary programming language. This experimental group was given the same set of programming exercises as the regular FORTRAN group, plus some additional programming assignments. One objective was to show that the students using ASPL could solve problems with fewer trials per problem than students using FORTRAN. This objective was essential in order to justify the cost-effectiveness of programming in ASPL. Another objective was to show that the students who had ASPL could be taught to master FORTRAN skills equivalent to the regular FORTRAN students in a short time. Some statistics relating the performance of the two groups are given below.

Activity	Experimental (26 students)	Regular (36 students)
Assign. #1		
Total Runs	28	65
Runs/Student	1.22	1.81
Assign. #2		
Total Runs	136	186
Runs/Student	5.23	5.16
Assign. #3		
Total Runs	143	220
Runs/Student	5.50	6.11
Assign. #4		
Total Runs	134	243
Runs/Student	5.15	6.75
Assign. #5		
Total Runs	201	302
Runs/Student	7.73	8.39
Assign. #6		
Total Runs	224	315
Runs/Student	8.62	8.75
Avg. # Completed Assignments	11 out of 12	7.4 out of 8
FORTRAN Final Exam Average	65.42	62.79

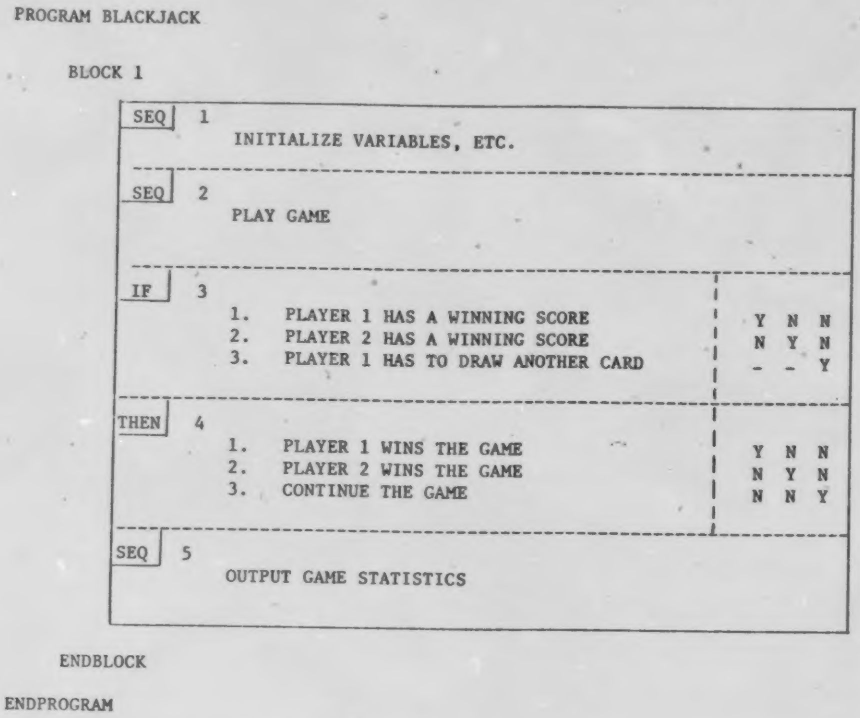
Because of the difficulty in setting up a meaningful experiment of this type, it is not possible to draw any hard conclusions from the above data. It can be observed, however, that the ASPL students generally solve the same problem as the FORTRAN students in fewer trials per problem. On the average the ASPL students solve more problems than the FORTRAN group. Perhaps the most impressive statistic is that the ASPL group was presented FORTRAN in only 2 weeks and obtained a higher average on the FORTRAN final exam than the regular FORTRAN group. The ASPL group also did the last two FORTRAN assignments both in ASPL and in FORTRAN. In general, the rating forms completed by students indicated a very positive attitude toward ASPL. Whether this language will become an alternative for teaching beginning programmers at MSU is uncertain at this time.

VIII. DISCUSSION AND CONCLUSIONS

Programming languages should be viewed only as tools to describe algorithms to computers. A programming language influences its users at least with respect to 1) the style of programming, 2) the conceptual understanding of how a problem can be solved by a computer, and 3) the range of problems which can be attacked by programming. The amount of time required for programmers to become fluent in a particular language should be minimal (i.e., 1-2 days). Additionally, a programmer should not have to routinely reference a programming manual in order to be functional in that language [2]. This suggests a need for languages which are simple, natural, and yet powerful. For most high-level languages, the emphasis has been on the power of the language. The language presented in this paper may serve as a model to illustrate that many desirable features can be incorporated into a relatively small language. ASPL was designed for beginning programmers, and therefore, it has the minimum basic constructs (sequential, DO-WHILE, and IF-THEN-ELSE). It attempts to minimize syntactical details in the coding of programs, thereby allowing the user to concentrate on the logic of a problem. Since many users' errors are associated with the logical flow of their programs, ASPL handles this by permitting only one entry and one exit point per BLOCK/SUBBLOCK, and by allowing the use of decision tables to simplify complex logic. These decision tables also provide an excellent means of documentation because they are easy for persons with varying technical backgrounds to read and construct. Furthermore, by changing entries in the decision table, one can modify an algorithm in a trivial manner. We believe that ASPL has the simplicity and intellectual appeal of a well-formed language and will generate among programmers the motivation to adopt new standards of programming practice.

REFERENCES

[1] M. V. Wikes, "Software Engineering and Structured Programming", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, Vol. SE-2, pp. 274-276, December 1976.
[2] H. D. Mills, "Software Development", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, Vol. SE-2, pp. 265-273, December 1976.
[3] D. J. Devine, "Decision Tables as a Basis of a Programming Language", DPMA QUARTERLY, Vol. 7, pp. 461-466, 1965.
[4] P. Freeman, "Toward Improved Review of Software Designs", PROCEEDINGS OF NATIONAL COMPUTER CONFERENCE, pp. 329-334, 1975.
[5] J. D. Gannon and J. J. Horning, "Language Design for Programming Reliability", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, Vol. SE-1, pp. 179-191, June 1975.
[6] N. Wirth, "On the Composition of Well-Structured Programs", COMPUTING SURVEYS, 6, 4, pp. 247-259, December 1974.
[7] H. Hughes, "A Tool Designed to Facilitate Structured Programming", BULLETIN OF SIGCSE, pp. 26-30, February 1977.
[8] Dahl, O. J., Dijkstra, E. W., Hoare, C. A. R., STRUCTURED PROGRAMMING, Academic Press, 1972.
[9] McDaniel, H., AN INTRODUCTION TO DECISION LOGIC TABLES, John Wiley, 1968.
[10] Chapman, A. E. and Callahan, M. A., "A Description of the Basic Algorithm Used in the DETAB/65 Preprocessor", COMMUNICATION ACM 10: 7, pp. 447-456, July 1967.
[11] Pooch, U. W., "Translation of Decision Tables", ACM COMPUTING SURVEYS 6: 2, pp. 125-249, June 1974.
[12] McDaniel, H., DECISION TABLE SOFTWARE, Brandon/System Press, 1970.
[13] Goos, Gerhard, "Programming Languages as a Tool in Writing System Software", SOFTWARE ENGINEERING, 1976.



```
PROGRAM BLACKJACK

    BLOCK 1

        SEQ  1
                INITIALIZE VARIABLES, ETC.

        SEQ  2
                PLAY GAME

        IF   3
                1.  PLAYER 1 HAS A WINNING SCORE          Y  N  N
                2.  PLAYER 2 HAS A WINNING SCORE          N  Y  N
                3.  PLAYER 1 HAS TO DRAW ANOTHER CARD     -  -  Y

        THEN 4
                1.  PLAYER 1 WINS THE GAME                Y  N  N
                2.  PLAYER 2 WINS THE GAME                N  Y  N
                3.  CONTINUE THE GAME                     N  N  Y

        SEQ  5
                OUTPUT GAME STATISTICS

    ENDBLOCK

ENDPROGRAM
```

Figure 2.
General Solution for BLACKJACK

```
PROGRAM BLACKJACK

  BLOCK 1

    SEQ 1
            1.  PLAYER1WIN=0
            2.  PLAYER2WIN=0
            3.  READ NUMGAMES
            4.  GAMECNT=0

    DO  2   WHILE GAMECNT < NUMGAMES

        DO  3   WHILE FLAG <> 1

            SEQ  4   1.  PLAY1TOTAL=PLAY1TOTAL+PLAY1CARD

            IF  5
                    1.  PLAY1TOTAL >= 17        |  N N - Y - -
                    2.  CARDCNT >= 5            |  N N Y N N N
                    3.  PLAY1TOTAL > 21         |  N N N - Y Y
                    4.  PLAY1CARD = 11          |  N Y - - Y -
                    5.  ACE > 0                 |  - - - - - Y

            THEN  6
                  1.  ACE=ACE+1                 |  N Y N N Y N
                  2.  PLAY1TOTAL=
                        PLAY1TOTAL-10           |  N N N N Y Y
                  3.  CARDCNT=CARDCNT+1         |  Y Y N N Y N
                  4.  PERFORM CARDVALUE         |  Y Y N N Y N
                  5.  FLAG=1                    |  N N Y Y N N
                  6.  ACE=ACE-1                 |  N N N N N Y
                  7.  PLAY1CARD=0               |  N N N N N Y

        SEQ  7      1.  PLAY2TOTAL=RANDOM NUMBER(17,25)
                    2.  PERFORM WINNER ROUTINE

    SEQ  8      1.  PERFORM CHAMPION ROUTINE

ENDBLOCK
```

Figure 3.
Refined Solution #1

```
SUBBLOCK DETWINNER

      +------------------------------------------------------------------+
      |IF  1                                         |                   |
      |                                              |                   |
      |          1.   PLAY1TOTAL > 21                |    N N N Y N      |
      |          2.   PLAY2TOTAL > 21                |    - - N - Y      |
      |          3.   PLAY1TOTAL > PLAY2TOTAL        |    Y - N - N      |
      |          4.   CARDCNT >= 5                   |    - Y - - -      |
      |------------------------------------------------------------------|
      |THEN  2                                       |                   |
      |                                              |                   |
      |          1.   PRINT "PLAYER1WON", PLAY1TOTAL |    Y Y N N Y      |
      |          2.   PRINT "PLAYER2WON", PLAY2TOTAL |    N N Y Y N      |
      |          3.   PLAY1WIN=PLAY1WIN+1            |    Y Y N N Y      |
      |          4.   PLAY2WIN=PLAY2WIN+2            |    N N Y Y N      |
      |                                              |                   |
      +------------------------------------------------------------------+

ENDBLOCK
```

Figure 4.
Refined Solution #2 for BLACKJACK

```
PROGRAM BLACKJACK
*//////////////////////////////////////////////////////////////////////////////
     BLOCK 1
          PLAYER1WIN=0
          PLAYER2WIN=0
          READ NUMGAMES
          GAMECNT=0
***THIS LOOP CONTROLS THE NUMBER OF GAMES PLAYED***
          DOWHILE GAMECNT  < NUMGAMES
               ACE=0
               GAMECNT-GAMECNT+1
               CARDCNT=0
               PLAY1TOTAL=0
               PLAY2TOTAL=0
               PLAY1CARD=INT(RANDOM*12+2)
               FLAG=0
***THIS LOOP DETERMINES PLAYER 1'S POINTS FOR EACH GAME***
                DOWHILE FLAG NE 1
                PLAY1TOTAL=PLAY1TOTAL+1
                     IF
                          PLAY1TOTAL GE 17;                N N - Y - -
                          CARDCNT GE     5;                N N Y N N N
                          PLAY1TOTAL GT 21;                N N N - Y Y
                          PLAY1CARD EQ  11;                N Y - - Y -
                          ACE GT         0;                - - - - - Y
                      THEN
                          ACE=ACE+1                        N Y N N Y N
                          PLAY1TOTAL=PLAY2TOTAL-10;        N N N N Y Y
                          PERFORM CARDVALUE;               Y Y N N Y N
                          ACE=ACE-1;                       Y Y N N Y N
                          PLAY1CARD=0;                     N N N N N Y
                          FLAG=1;                          N N Y Y N N
                     ENDIF

             ENDDO
```

Figure 5. (Continued on next page)
ASPL Program for BLACKJACK

```
    ***NOW GENERATE THE DEALER'S SCORE***
                   PLAY2TOTAL=INT(RANDOM*8+17)
                   PERFORM DETWINNER
              ENDDO
              PERFORM CHAMPION
          ENDBLOCK


*//////////////////////////////////////////////////////////////////////////////
          SUBBLOCK DETWINNER
               IF
                    PLAY1TOTAL > 21;                    N N N Y N
                    PLAY2TOTAL > 21;                    N - N - Y
                    PLAY1TOTAL > PLAY2TOTAL;            Y - N - N
                    CARDCNT GE    5;                    - Y - - -
                THEN
                    PRINT "PLAYER1WON", PLAY1TOTAL;     Y Y N N N
                    PRINT "PLAYER2WON", PLAY2TOTAL;     N N Y Y Y
                    PLAY2WIN-PLAY1WIN+1;                Y Y N N N
                    PLAY2WIN=PLAY2WIN+1;                N N Y Y Y
               ENDIF
          ENDBLOCK
*//////////////////////////////////////////////////////////////////////////////
          SUBBLOCK CHAMPION
               IF
                    PLAT1WIN GT PLAY2WIN;               Y
               THEN
                    PRINT "PLAYER 1 IS CHAMP";          Y
                    PRINT "GAMES WIN", PLAY1WIN;        Y
               ELSE
                    PRINT "PLAYER 2 IS CHAMP"
                    PRINT "GAMES WON", PLAY2WIN
               ENDIF
          ENDBOOCK
*//////////////////////////////////////////////////////////////////////////////
          SUBBLOCK CARDVALUE
               PLAY1CARD=INT(RANDOM*12+2)
               IF
                   PLAY1CARD > 9 AND PLAY1CARD   14;    Y
               THEN
                   PLAY1CARD=10;                        Y
               ENDIF
          ENDBLOCK
      ENDPROGRAM
```

Figure 5. (Continued)
ASPL Program for BLACKJACK

Initial Evaluation Results for an Introductory Programming Course Without Lectures

Margret Hazen*, Cecilia Daly, David W. Embley, George Nagy, and William Prange
Department of Computer Science
University of Nebraska-Lincoln

Introduction

Soaring enrollments, tight budgets, high drop-out rates, and wide diversity in quality and content of traditional classroom instruction of introductory computer programming courses lead to a search for better methods of teaching at a lower cost. From observation, introductory programming students seem to learn best from direct computer feedback on programs submitted for execution, from clearly written, self-explanatory examples, and from discussing a given problem or exercise one-to-one with instructors, with teaching assistants, and with other students. Accordingly, a method of instruction was sought that would maximize the students' exposure to these learning resources and that would also meet the demands of enrollment, budget, and student satisfaction. The individualized lectureless teaching strategy developed at the University of Nebraska-Lincoln emerged from this need.

Instructional theorists have long argued that learning takes place most effectively in meaningful problem-solving situations that involve the learner to a maximum degree (Bruner, 1961; Dewey, 1938). If this type of learning is to be accomplished, an individualized approach to instruction appears necessary. Cross (1976) has outlined five criteria of successful individualized instruction, and all but one of these are characteristics of UNL's lectureless strategy of instruction.

1. Responsibility and Participation. Individualized instruction demands that the students take on far more responsibility for their own learning than do more traditional methods. This characteristic is intertwined with the need for active participation by the learners to enable them to achieve maximum learning. The lectureless method was developed expressly to accomplish these ends since programming languages and problem-solving skills require the active participation of the student for learning to occur. With this instructional method, students must initiate their own problem-solving steps; with the lecture method, students can rely more on the instructor and be more passive in their learning.
2. Learning Objectives. Individualized instruction requires a clear statement of the competencies to be achieved. Students are more likely to gain these competencies if they are stated explicitly. Lectureless students were provided with both a course guide that specified course objectives, and weekly assignment sheets that reiterated the objectives for that assignment. Such explicitly stated objectives were not (and frequently are not) provided in the lecture mode of instruction.
3. Small Units of Instruction. Small lesson units of instructional modules facilitate individualized instruction since they allow a student to grasp more easily the essence of what has to be accomplished. The lectureless students received more assignments, each one more narrowly defined, than did the lecture students. Smaller instructional units also provide more opportunities for feedback and reinforcement.
4. Feedback and Reinforcement. Feedback provides the steering mechanism that helps students achieve learning goals, while reinforcement accelerates progress towards goals. The lectureless students completed weekly assignments and took oral quizzes to test their understanding of that week's objectives. Feedback and reinforcement specific to each student were key features of these oral quizzes.
5. Self-Pacing. Although self-pacing is

---

*The principal author has taught in the Computer Science Department but is currently completing a dissertation in the College of Education.

a desirable characteristic of individualized instruction, past experience had indicated a need for a common time table. Students can learn from each other during the problem-solving process, and common problems with the same due date facilitate this learning. Also, falling behind in programming assignments is far more detrimental to the student than is falling behind in reading assignments. According to past experience, students who do not keep up are unlikely ever to catch up.

Other attempts at lectureless programming instruction are few. UNL's approach is similar in form to the University of Minnesota's approach (Couger, 1978), but differs with respect to self-pacing from Michigan State University's and the University of California's approach (LeCureaux and Burnett, 1977; White and Pessel, 1974).

This paper reports on the effectiveness of UNL's individualized lectureless method of instruction for introductory FORTRAN programming classes for non-computer science majors. Although effectiveness is to be measured by several dimensions (Nagy, 1978), only three measurement results are currently available. Preliminary observations have already been presented (Daly, Embley, and Nagy, 1979) and more comprehensive results will be published at a later date.

The measurements reported on here are the final examination grades, the course evaluation questionnaires, and the computer attitude questionnaires. These instruments were used to help determine 1) whether lectureless students would learn as much or more than the lecture students, 2) whether lectureless students would be as satisfied or more satisfied with their learning experience, and 3) whether lectureless students would have an equally favorable attitude towards computers and computer programming.

Description of the Lectureless Method

At the end of the Fall Semester, 1978, three lecture and four lectureless sections of CS 237, FORTRAN Programming for Business, were evaluated. Since the actual development and implementation of the teaching strategy for these courses have been detailed elsewhere (Nagy, 1978; Daly, Embley, and Nagy, 1979), only the key features are outlined here.

The content of the lecture and lectureless sections was standardized as much as possible by using the same text (Lipshutz and Poe, 1978). Programming assignments were identical for all four lectureless sections and were comparable to those of the lecture sections in that they had the same objectives but placed the problems in different contexts.

The differences between instructional methods lay in the nature of the instructional information and assistance that students were provided. Lecture students were guaranteed two hours a week of lectures and instructor-class interaction. Individualized assistance was available if the student sought out the instructor or one of the consultants available to the general programming population. Such assistance, however, was not sought out by all students.

Students in the lectureless sections did not have formal large group meetings except for an orientation session and two review sessions before the mid-semester and final examinations. Course information was provided by.

1) the course guide — a comprehensive, forty-page booklet containing a course outline, course objectives, available learning resources, and course mechanics.

2) weekly assignment sheets — containing assignment objectives, the reading assignment, exercise problems, and the actual programming assignment. These sheets provided an average of five pages of information.

3) a bulletin board — located in course headquarters for general messages, posted problem solutions, and quiz sign-up sheets.

Individualized assistance was provided to the lectureless students via:

1) weekly oral quizzes — a one-to-one evaluation by an instructor or teaching assistant of the student's learning which also provided a means to clarify misunderstandings and to set another appointment for more assistance, if necessary.

2) teaching assistant consultants — scheduled to be in the course headquarters for thirty hours per week. These consultants were prepared to answer questions the lectureless students encountered.

3) course headquarters — a room specifically set aside and made comfortable for students to encourage use of this room for study and group discussion.

Lectureless students accessed the computer through SIMPLE, a text editor designed for beginners and for users with minimal system requirements (Embley and Nagy, 1978) and implemented specifically for use in the lectureless course. The lecture students used CMS (IBM, 1977).

Evaluation Design and Subjects

The individualized lectureless method

of instruction was compared to the traditional lecture method of instruction in a straightforward experimental design. Students were effectively randomly assigned to four lectureless sections and three lecture sections since they had registered for these sections six months prior to their designation as either lecture or lectureless, and each section was filled to capacity at the time of registration. No students changed sections across instructional method, thereby ruling out self-selection effects.

Student data for the lectureless sections was grouped together for analytic purposes since all these students interacted with all three instructors for consultation and for oral quiz evaluations. Two of the three lecture sections were taught by the same instructor and were taught in a manner as comparable as possible for the evaluation. Programming assignments and concepts differed only slightly from the lectureless sections. The third lecture section was eliminated from the final examination data analysis since students in these sections were given different assignments, and the final examination did not count for the same proportion of the grade as for the other sections. Course evaluation and attitude data was considered to be unaffected, and therefore all three sections were grouped together for these analyses.

A total of 193 University of Nebraska students were enrolled in the seven sections evaluated; 108 were enrolled in the lectureless sections, and 85 were enrolled in the lecture sections. A background survey indicated that the students represented a fairly homogeneous group. Eight-four percent were business majors. Sixty-five percent of the students had completed sixty hours or less, indicating a large freshman and sophomore population. Also, most of the students (seventy-seven percent) were between eighteen and twenty-one years of age.

Evaluation Instruments

The final examination scores represent a cognitive measure of effectiveness. They were designed by all three participating instructors. Three versions were given randomly to the students in the examination rooms. Questions for these three versions were analogous — presenting the same problems in different contexts.

The course evaluation and computer attitude questionnaires represented affective or attitudinal measures. They were designed by the principal author with assistance from the other authors. The course evaluation forms consisted of two versions--lecture and lectureless. Seven questions actually differed due to the nature of instruction. One-half of the questions on the computer attitude questionnaire were about attitudes toward computer programming and the other half were questions about computers in general. Both the course evaluation and the computer attitudes questionnaire were composed of a balanced mixture of positively and negatively phrased questions.

Results

The analysis of variance results for the final examination scores are shown in Table 1.

Table 1 Final Examination Data (N=138)

	X	SD	N	$F_{(1,136)}$
Lecture	68.5	14.2	45	.771
Lectureless	66.1	15.4	93	p<.38
			138	

Examination scores represent the number of points out of a possible score of 90. (Statistics were produced using SPSS subprograms (Nie, et.al., 1975).)

The results for the course evaluations are shown in Table 2.

Table 2 Course Evaluation Data (N=160)

	X	SD	N	$F_{(1,158)}$
Lecture	126.7	11.2	67	3.724
Lectureless	130.4	11.9	93	p<.06
			160	

The course evaluation mean represents a total score of all the Likert five-point scale responses after negatively phrased questions were reversed. The results of an item analysis for the course evaluation appear in Appendix A, superimposed on a sample course evaluation form.

The results of the attitude questionnaires are shown in Table 3.

Table 3. Attitude Data (N=155)

	Mean Before	Mean After
Lecture	X=45.8	44.8
	SD=(5.0)	(5.0)
Lectureless	45.4	44.5
	(5.2)	(5.2)

The attitude questionnaire mean represents a total score of all the Likert five-point scale responses after negatively phrased questions were reversed. An analysis of variance test indicated no significant differences for students in the lecture and lectureless sections either at the beginning of the semester or at the end of the semester. The results of an item analysis for the attitude questionnaires appear in Appendix B, superimposed on a sample computer attitude questionnaire.

Discussion

The final examination scores indicate that the same content was learned under the lecture and lectureless methods. However, students in the lectureless sections spent less time (8.3 hours per week) than did the lecture students (7.9 hours per week plus two hours of class time and transportation time) to accomplish the same amount of learning. This reduction of time spent on the course is considered an advantage of the lectureless mode of instruction, particularly since time seems to be the predominant concern of students in the course as indicated by the responses to the open ended questions on the course evaluation form.

Attitudes towards computer programming and computers also do not differ significantly for the lecture and lectureless students. Attitudes did however decline from the beginning of the semester to the end; a change that was unexpected but not totally surprising since computer programming is often a frustrating and difficult task to master--especially at the introductory stages.

Course evaluation results indicate that the students were more satisfied with the lectureless than the lecture method although not significantly so at the .05 level.

An exploratory analysis of the course evaluation items indicates several questions differed significantly (at less than the .05 level) for the lectureless and the lecture sections. Questions 4 and 13, for example ($F_{1,158}=11.39$, $p<.001$ and $F_{(1,158)}=8.03$, $p<.005$), indicate that the lectureless students, more so than the lecture students, felt that the midterm and quizzes accurately reflected what they knew. Lectureless students also felt that course objectives were stated more clearly (question 10, $F_{(1,158)}=16.19$, $p<.001$). Since quizzes and midterms were based explicitly on the course objectives for the lectureless students, the responses to these questions may indicate that students do indeed gain competencies more readily if they are stated explicitly in terms of objectives.

The results of question 5 ($F_{(1,158)}=3.79$, $p<.05$) indicate that lecture students felt that instructors found ways to help students answer their own questions more than the instructors and teaching assistants did in the lectureless sections. This could be because the students did not get to know one instructor over the semester in the lectureless sections and were therefore exposed to a greater variety, and in a sense, less personal contacts with instructors and teaching assistants.

The results of question 6 ($F_{(1,158)}=6.22$, $p<.01$) are gratifying to the participants in the project. Students in the lectureless sections did seem to find SIMPLE easier to learn than the CMS text editor of the lecture sections.

The first 36 items of the course evaluation form (Appendix A) were used as an aggregate measure of attitudes towards the course. Additional questions referred to the form itself, the number of hours spent on homework, and general opinions towards the course. Only 26 students, or sixteen percent of the total sample, felt that the form did not permit them to describe their impressions of the course accurately. To allow for this possibility, two open-ended questions were included so that students could express opinions that were not covered by specific items on the form. Comments from these open-ended questions as well as the results of in-depth interviews conducted with twenty students from both groups corroborate the overall satisfaction with the lectureless mode.

An earlier progress report (Daly, Embley, and Nagy, 1979) also substantiates the results of this paper. Midterm grades for those sections that gave identical midterms differed significantly ($p<.05$), with the lectureless students obtaining higher scores. Drop rates were lower for the lectureless than for the lecture sections.

As more sections are taught in the lectureless mode and as the faculty assigned to the lectureless sections are replaced with graduate assistants for evaluation and consultation purposes, the department foresees a reduction in course delivery costs. Cost reductions were not attained for the pilot version since extra personnel were committed to the lectureless course as a precautionary fallback measure.

Further analyses of drop-out and attrition rates as well as the effect of a personality measure, locus of control, will provide additional information for the understanding of how the lectureless mode aids or fails students in their learning efforts.

References

1. Bruner, J.S. The Process of Education, Cambridge: Harvard University Press, 1961.
2. Couger, J.D. ed. "Comparing COBOL Pedagogy", Computing Newsletter, Colorado Springs: University of Colorado, December, 1978.
3. Cross, K.P. Accent on Learning, San Francisco: Jossey-Bass, 1976.
4. Daly, C., Embley, D.W., and Nagy, G. "A Progress Report on Teaching Programming to Business Students Without Lectures", Proceedings ACM-SIGCSE, Dayton, Ohio, 1979.
5. Dewey, J. Experience and Education, New York: Macmillan, 1938.
6. Embley, D.W. and Nagy, G. "SIMPLE Specifications", Department of Computer Science, University of Nebraska-Lincoln, 1978.
7. IBM, Virtual Machine Facility/370: CMS User's Guide, Poughkeepsie, New York: IBM, 1977.
8. LeCureaux, F. and Burnett, J. "Modularized FORTRAN Programming", Department of Computer Science, Michigan State University, 1977.
9. Lipschutz, S. and Poe, A. Theory and Problems of Programming with FORTRAN, New York: McGraw-Hill, 1978.
10. Nagy, G. "Introductory Programming Courses Without Lectures", Proceedings - CCUC, Denver, Colorado, 1978.
11. Nie, N.H., Hull, C.H., Jenkins, J.G., Steinbrenner, K. and Bent, D.H., eds. SPSS: Statistical Package for the Social Sciences, 2nd ed., New York: McGraw-Hill, 1975.
12. White, R.M. and Pessel, D. "A Self-Paced Course in Introductory FORTRAN Programming", Department of Electrical Engineering and Computer Sciences, University of California - Berkeley, 1974.

APPENDIX A

Section 3, 5, 7, & 9

LECTURELESS

Name______________________

Section______________________

This form represents your final opportunity to evaluate CS 237. As with the other forms, your anonymity will be preserved. Only summary information will be made available to the instructor.

Please circle the number that most accurately reflects how you feel about the statements to the left. Feel free to add comments if the question is unclear.

		Percentages				
		STRONGLY DISAGREE	DISAGREE	NEITHER AGREE NOR DISAGREE	AGREE	STRONGLY AGREE
MEAN (N=93)		(1)	(2)	(3)	(4)	(5)
3.67	1. The programming assignments took too much time to complete	2.2	17.2	15.1	43.0	22.6
2.11	2. The text was not useful for completing the programming assignments	21.5	54.8	16.1	6.5	1.1
3.45	3. The consultants in Ferguson 17 were available when I needed them	4.3	21.5	15.1	43.0	16.1
3.71	4. The results of the midterm accurately reflected what I knew	1.1	18.3	10.8	48.4	21.5
3.46	5. The instructors and teaching assistants found ways to help students answer their own questions	1.1	16.1	21.5	58.1	3.2
2.01	6. The SIMPLE system was difficult to learn	23.7	59.1	9.7	7.5	0
2.57	7. There was no real concern for my progress in this course	4.3	47.3	35.5	12.9	0
2.30	8. I just tried to finish the assignments rather than to learn	11.8	61.3	12.9	12.9	1.1
3.07	9. This course was more difficult than other college courses I have taken	4.3	30.1	26.9	32.3	6.5
3.93	10. The objectives of the course were clearly stated	1.1	1.1	10.8	78.5	8.6
3.70	11. The Course Guide was useful for this course	0	10.8	17.2	63.4	8.6
3.01	12. I tended to procrastinate my work in this course	4.3	34.4	23.7	31.2	6.5
2.37	13. The results of the quizzes did not accurately reflect what I knew	8.6	62.4	16.1	9.7	3.2
3.52	14. The SIMPLE system worked well	2.2	10.8	30.1	47.3	9.7
4.15	15. This course demanded self-discipline	0	2.2	1.1	76.3	20.4
2.83	16. If my schedule permits it, I would like to take another programming class	19.4	23.7	19.4	30.1	7.5
3.81	17. I found working with other students on my assignments very useful	2.2	10.8	10.8	57.0	19.4
1.98	18. I did not feel free to ask questions of the instructors or the teaching assistants	28.0	58.1	4.3	7.5	2.2
3.94	19. The quizzes demanded I think clearly and carefully	0	2.2	17.2	65.6	15.1
1.90	20. This course was a waste of time	30.1	54.8	10.8	3.2	1.1

APPENDIX A

Section 3, 5, 7, & 9

LECTURELESS

			STRONGLY DISAGREE	DISAGREE	NEITHER AGREE NOR DISAGREE	AGREE	STRONGLY AGREE
			(1)	(2)	(3)	(4)	(5)
2.37	21.	I would not recommend this course to a friend with similar interests and abilities as mine	12.9	54.8	17.2	12.9	2.2
1.94	22.	The programming assignments were not valuable for the understanding of the course material	18.3	72.0	7.5	2.2	0
3.67	23.	Room 17 (Course Headquarters) proved to be a very convenient place to work	0	14.0	17.2	57.0	11.8
2.04	24.	The grading policies were not fair	17.2	65.6	12.9	4.3	0
3.23	25.	The completion of the programming assignments required the assistance of another person	3.2	30.1	14.0	46.2	6.5
3.72	26.	I enjoyed the increased time and flexibility to work on my assignments	3.2	11.8	11.8	55.9	17.2
3.51	27.	The level and style of the text were appropriate	3.2	14.0	16.1	62.4	4.3
3.79	28.	Room 17 (Course Headquarters) was a convenient place to get help with my programming assignment	0	9.7	11.8	68.8	9.7
3.97	29.	My responsibilities in this course were clearly defined	0	2.2	5.4	86.0	6.5
3.47	30.	The instructors and teaching assistants adapted to my level of comprehension	1.1	12.9	29.0	51.6	5.4
3.79	31.	The teaching assistants were helpful	1.1	7.5	15.1	64.5	11.8
3.54	32.	The oral quizzes were a useful learning aid	4.3	7.5	24.7	57.0	6.5
2.45	33.	The examples in the text were not very useful	8.6	57.0	18.3	12.9	3.2
3.43	34.	This course increased my interest in the subject matter	3.2	15.1	25.8	47.3	8.6
2.12	35.	The midterm did not demand that I think clearly and carefully	11.8	71.0	11.8	4.3	1.1
2.42	36.	This course offered too much of an opportunity for cheating	12.9	51.6	18.3	15.1	2.2
3.30	37.	The questions on this form permitted me to describe my impressions of this course accurately	2.2	16.1	35.5	41.9	4.3
2.91	38.	The film (Ultimate Machine) held my attention	7.5	28.0	41.9	21.5	1.1
2.69	39.	The tour held my attention and was very informative	12.9	26.9	39.8	19.4	1.1
8.29	40.	For this course the number of hours per week I spent on homework	(3) 10.8	(6) 32.3	(9) 34.4	(12) 15.1	(15) 7.5

41. This course needed
- a) two lectures a week
- b) one lecture a week
- c) one lecture every other week
- d) was fine with no lectures
- e) other, please specify ______________

42. What was the most positive aspect of this course for you?

43. What was the most negative aspect of this course for you?

APPENDIX A

Section 1, 4, & 8

LECTURE

Name______________________

Section____________________

This form represents your final opportunity to evaluate CS 237. As with the other forms, your anonymity will be preserved. Only summary information will he made available to the instructor.

Please circle the number that most accurately reflects how you feel about the statements to the left. Feel free to add comments if the question is unclear.

		Percentages				
MEAN (N=67)		STRONGLY DISAGREE (1)	DISAGREE (2)	NEITHER AGREE NOR DISAGREE (3)	AGREE (4)	STRONGLY AGREE (5)
3.54	1. The programming assignments took too much time to complete	1.5	13.4	23.9	52.2	9.0
2.33	2. The text was not useful for completing the programming assignments	9.0	64.2	13.4	11.9	1.5
3.22	3. The consultants in Ferguson 21 were available when I needed them	6.0	19.4	23.9	47.8	3.0
3.15	4. The results of the midterm accurately reflected what I knew	3.0	29.9	23.9	35.8	7.5
3.72	5. The instructor found ways to help students answer their own questions	0	7.5	25.4	55.2	11.9
2.33	6. The CMS system was difficult to learn	6.0	68.7	11.9	13.4	0
2.52	7. There was no real concern for my progress in this course	3.0	53.7	31.3	11.9	0
2.36	8. I just tried to finish the assignments rather than to learn	4.5	68.7	17.9	4.5	4.5
3.16	9. This course was more difficult than other college courses I have taken	4.5	25.4	25.4	38.8	6.0
3.46	10. The objectives of the course were clearly stated	3.0	13.4	20.9	59.7	3.0
3.54	11. The lectures explained the course material clearly	4.5	11.9	20.9	50.7	11.9
2.85	12. I tended to procrastinate my work in this course	6.0	32.8	34.3	23.9	3.0
2.79	13. The results of the quizzes did not accurately reflect what I knew	4.5	41.8	29.9	17.9	6.0
3.61	14. The CMS system worked well	1.5	9.0	19.4	67.2	3.0
4.02	15. This course demanded self-discipline	0	0	10.4	77.6	11.9
2.99	16. If my schedule permits it, I would like to take another programming class	11.9	25.4	19.4	38.8	4.5
3.55	17. I found working with other students on my assignments very useful	4.5	9.0	31.3	37.3	17.9
1.99	18. I did not feel free to ask questions of the instructor	28.4	55.2	9.0	4.5	3.0
3.84	19. The quizzes demanded I think clearly and carefully	0	1.5	20.9	70.1	7.5
1.85	20. This course was a waste of time	29.9	58.2	10.4	0	1.5

Section 1, 4, & 8

LECTURE

			STRONGLY DISAGREE (1)	DISAGREE (2)	NEITHER AGREE NOR DISAGREE (3)	AGREE (4)	STRONGLY AGREE (5)
2.19	21.	I would not recommend this course to a friend with similar interests and abilities as mine	16.4	59.7	13.4	9.0	1.5
1.78	22.	The programming assignments were not valuable for the understanding of the course material	28.4	67.2	3.0	1.5	0
2.36	23.	The lectures were tiring and dull	7.5	58.2	25.4	9.0	0
2.03	24.	The grading policies were not fair	19.4	62.7	13.4	4.5	0
2.97	25.	The completion of the programming assignments required the assistance of another person	4.5	34.3	23.9	34.3	3.0
3.03	26.	The lectures introduced stimulating ideas about the subject	1.5	26.9	40.3	29.9	1.5
3.63	27.	The level and style of the text were appropriate	0	9.0	22.4	65.7	3.0
3.42	28.	The lectures summarized material in a manner which aided retention	3.0	10.4	32.8	49.3	4.5
3.82	29.	My responsibilities in this course were clearly defined	0	7.5	11.9	71.6	9.0
3.64	30.	The instructor adapted to my level of comprehension	0	9.0	25.4	58.2	7.5
3.55	31.	The lectures demonstrated the importance and significance of the subject matter	0	11.9	25.4	58.2	4.5
3.52	32.	The lectures made it clear how each topic fit into the course	1.5	11.9	22.4	61.2	3.0
2.25	33.	The examples in the text were not very useful	11.9	62.7	16.4	6.0	3.0
3.39	34.	This course increased my interest in the subject matter	4.5	10.4	31.3	49.3	4.5
2.06	35.	The midterm did not demand that I think clearly and carefully	14.9	65.7	17.9	1.5	0
2.67	36.	This course offered too much of an opportunity for cheating	9.0	41.8	28.4	14.9	6.0
3.40	37.	The questions on this form permitted me to describe my impressions of this course accurately	0	13.4	35.8	47.8	3.0
7.93	38.	For this course the number of hours per week I spend on homework outside of classtime was	(3) 10.4	(6) 38.8	(9) 31.3	(12) 14.9	(15) 4.5

39. What was the most positive aspect of this course for you?

40. What was the most negative aspect of this course for you?

APPENDIX B

Name____________________

Social Security #____________________

Section____________________

	Before	After
Lecture	N=65	65
Lectureless	N=90	90

COMPUTER SURVEY

Please circle the number that most accurately reflects how you feel about the statements to the left.

		PERCENTAGES				
MEAN		STRONGLY DISAGREE (1)	DISAGREE (2)	NEITHER AGREE NOR DISAGREE (3)	AGREE (4)	STRONGLY AGREE (5)
4.3 4.1 4.0 3.9	1. Computer programming is an important skill to have for future job opportunities	0 0 2.2 2.2	1.5 1.5 4.4 6.7	6.2 15.4 13.3 8.9	49.2 58.5 54.4 64.4	43.1 24.6 25.6 17.8
2.2 2.5 2.2 2.5	2. I try to avoid working on my computer programs until the last minute. They are my least favorite assignments.	20.0 13.8 25.6 17.8	52.3 44.6 45.6 42.2	18.5 21.5 18.9 18.9	4.6 13.8 7.8 17.8	4.6 6.2 2.2 3.3
1.7 1.9 1.8 2.0	3. Computers are not very useful because they are always breaking down or making mistakes	41.5 26.2 37.8 23.3	49.2 60.0 50.0 58.9	7.7 10.8 5.6 12.2	0 3.1 3.3 2.2	1.5 0 3.3 3.3
3.8 3.7 3.9 3.6	4. Computers are fascinating and exciting to work with	3.1 0 0 0	6.2 6.2 1.1 6.7	7.7 24.6 22.2 31.1	73.8 60.0 64.4 56.7	9.2 9.2 12.2 5.6
2.6 2.5 2.6 2.3	5. I become frightened and panicky at the thought of having to write a computer program	12.3 7.7 15.6 14.4	36.9 43.1 37.8 55.6	29.2 41.5 21.1 16.7	20.0 7.7 18.9 10.0	1.5 0 6.7 3.3
2.2 2.3 2.1 2.3	6. I will try to avoid working with computers in my future job	10.8 10.8 16.7 12.2	64.6 53.8 61.1 55.6	20.0 29.2 18.9 21.1	3.1 6.2 2.2 8.9	1.5 0 1.1 2.2
3.5 3.6 3.6 3.7	7. Computers will help to bring about a better way of life for the average person	1.9 0 1.1 1.1	3.1 6.2 4.4 6.7	43.1 30.8 34.4 24.4	46.2 58.5 52.2 56.7	6.2 4.6 7.8 11.1
1.8 2.1 2.0 2.0	8. Computer programming seems to be a fairly useless skill	33.8 21.5 28.9 26.7	58.5 63.1 58.9 56.7	0 1.5 2.2 6.7	6.2 12.3 7.8 6.7	1.5 1.5 2.2 3.3
3.9 4.0 3.9 3.9	9. Computers are extremely accurate, efficient, and reliable	1.5 0 0 1.1	6.2 1.5 5.6 4.4	13.8 20.0 18.9 15.6	58.5 55.4 51.1 61.1	20.0 23.1 24.4 17.8
3.1 2.8 3.0 2.8	10. I always look forward to working on my computer programming assignments	6.2 9.2 6.7 7.8	13.8 23.1 23.3 32.2	46.2 44.6 37.8 34.4	30.8 21.5 26.7 21.1	3.1 1.5 5.6 4.4
4.1 4.0 4.0 3.9	11. Computers are important for the efficient operation of large and small businesses	0 0 0 0	1.5 3.1 5.6 5.6	9.2 10.8 12.2 12.2	69.2 72.3 61.1 66.7	20.0 13.8 21.1 15.6
2.3 2.1 2.3 2.2	12. I hesitate working with computers because they are strange and frightening	10.8 13.8 14.4 16.7	55.4 67.7 52.2 56.7	23.1 12.3 20.0 20.0	10.8 6.2 12.2 4.4	0 0 1.1 2.2

Approaches to Lesson Distribution and Delivery

Chaired by Richard E. Pogue
Medical College of Georgia

CONDUIT: An Organized System for Dissemination of Quality Computer-Based Curriculum Materials, James W. Johnson, CONDUIT

EDUNET: Sharing of Curriculum Materials By Means of a Computer Network with Multiple Hosts, Elizabeth R. Little, EDUNET

A Microcomputer-Based Approach to Lesson Distribution and Delivery, Bodie Marx, Milliken Communications Corporation

A System- and Language-Independent Approach to Lesson Creation and Distribution, Richard E. Pogue, Medical College of Georgia

This session presents four different approaches to the problems of CAI lesson distribution and delivery. While each approach has an implicit or explicit approach to authoring, this aspect of the lesson authoring-distribution-delivery process will be discussed only in passing. Each presentation will focus on the questions: What is the approach?; Why is this approach being used?; What are the experiences to date?; and What is the future of this approach apt to be?

CONDUIT represents a systematic approach to the DISTRIBUTION of quality, peer-reviewed lesson material. Materials submitted by authors for possible distribution by CONDUIT are subjected to a thorough review of subject matter, instructional quality, and ease of use. Standards have been established for design, programming, packaging for distribution, and peer review. Only materials that meet CONDUIT standards are accepted for distribution. CONDUIT has recently decided to consider for distribution materials written in CAI programming languages, a change from its former policy which limited consideration to materials written in BASIC and FORTRAN.

EDUNET, on the other hand, is primarily a DELIVERY mechanism, with distribution implicit in the operation of its multiple-host network. EDUNET'S mission is to provide access to materials on the computers in the EDUNET network. It serves as the marketing arm for its host suppliers; provides facilitating information and user and administrative services; and supports development of some network products. In the role of coordinator and facilitator, EDUNET exercises no control over the quality of materials available through its network, nor does it have any involvement in the authoring process.

The approach taken by Milliken Communications Corporation, a unit of a for-profit publishing company, concentrates primarily on the AUTHORING and DISTRIBUTION of peer-reviewed lessons, with the unique aspect that delivery is on a standard low-cost microcomputer widely available for use in homes and schools. Milliken selects content areas for which they believe lessons are needed, seeks out authors, requires adherence to standards for lesson design and development, and provides lesson entry and testing services. After peer review, lessons are announced for distribution on floppy disks.

The last approach involves the use of computer-based mechanisms to facilitate AUTHORING and DISTRIBUTION. The authoring process uses a customizable prompting program for content entry and produces a language-independent lesson content file. Translator programs convert the lesson content file into operating computer programs. The objectives of this approach are two-fold: to permit authors to create lesson materials without having to learn a computer language, and to facilitate distribution by providing lessons that can be run on different types of computers using different computer and CAI languages.

A COMPUTER LOAN SERVICE

Con Dietz
Gene A. Kemper
University of N.D.

A COMPUTER LOAN SERVICE

by

Con Dietz and Gene A. Kemper
Computer Center
University of North Dakota

During the late 1960s and early 1970s there was a growing interest among secondary schools in North Dakota regarding using computers in classrooms. Several of these schools looked to the University of North Dakota Computer Center to fulfill a service and leadership role by providing some type of temporary computing capability to such schools. North Dakota, being a sparsely poplated (600,000 plus people in a state approximately 350 miles east-west by 250 miles north-south) rural state, provided little opportunity for students to be exposed to any type of computer facility. Furthermore, school administrators and teachers generally lacked experience with computers.

The Computer Center proposed to acquire a PDP-8 minicomputer and loan this system to seven secondary schools each school year for a period of approximately one month. Because the University administration felt obligated to provide a service to the state, the Computer Center was funded to purchase a minicomputer. The Computer Loan Service (CLS) was thus born in 1973.

Specifically, the CLS purchased a 4K(words) PDP-8F with EDUSYSTEM 5, an ASR33 teletype and approximately $100 in DEC materials (manuals and texts). Other materials (such as the Huntington manuals and CAMP series) were added. The monthly cost to a participating school renting this system was determined by the number of students in the high school (grades 9-12). The cost was 70 cents per high school student with a minimum cost of $140 and a maximum cost of $280. At the time the CLS became operational there were approximately 250 elementary/secondary schools in the state of which 131 had less than 100 students, 69 had between 100 and 200 students, 32 had between 200 and 400 students, and only 18 had enrollments of over 400. Thus, most of the schools paid the minimum charge.

There were other costs involved. Typically a school needed to pay the mileage costs for a University vehicle to deliver and pick up the computer system. This cost is currently 15 cents per mile. Although the school was required to pay per diem (food/lodging) costs for the Computer Center staff member to deliver and pick up the computer system, no charges were made for personnel time. Programming manuals were available at a cost of 50 cents each. Most schools paid between $5 and $10 for teletype paper and tape.

The only other cost to a participating school was to support the attendance of at least one teacher at a day-and-a-half workshop held on the University of North Dakota campus in late August or early September just after the beginning of the school year. This workshop provided instruction in the essentials of BASIC and the operation of the computer, primarily loading EDUSYSTEM 5. No registration fee was charged for the workshop.

These charges were determined such that the CLS would be self-supporting (except for personnel costs). The charges provided funds for repair, replacement and addition of manuals and texts, and a fund for possible equipment expansion. If the computer system breaks down, the travel charges and repair costs are the responsibility of the CLS. Furthermore, the school at which the problem occurs is charged for only that portion of the month during which the system was operating. If the system needs to be removed from the school for repair it is returned to that school if the repair is completed before the scheduled period ends; otherwise the system is delivered to the next school. In other words, the schedule of the system is not altered to accommodate repair time.

For the CLS to be effective it was necessary to actively market its services. Initially the CLS sought the assistance of the Mathematics and Science Division of the State Department of Public Instruction for suggestions of schools most likely to participate. Letters were sent to the superintendents of the suggested schools. Seven schools filled the available periods that first year. During the first year of operation a news release was issued by the University News Bureau to newspapers in the area of each participating school. This release described the CLS and named the teachers in the local school who would be using the computer.

During each of the following years of operation the principal advertising medium has been the Mathematics/Science Newsletter from the State Department of Public Instruction. A description of the CLS services typically is in the January issue of the Newsletter. The description contains a commitment form to be filled out by schools wishing to participate during the next school year. These forms are returned to the CLS by mid-March. At that time each participating school is requested to specify first, second, and third choices for the scheduled available time periods. A final schedule and workshop date is sent each school in late April. This timing permits the teachers to plan more effectively the use of the computer system. The final schedule and workshop date is accompanied by a programming/computer operations manual which contains the material to be covered in the workshop.

The CLS has been fortunate to have been invited to describe its services at statewide meetings of both the School Boards Association and the Association of Superintendents and Administrators as well as on TV and radio newscasts.

During the first year of operation it became immediately evident that 4K (words) of memory was insufficient. Thus, another 4K was installed (with EDUSYSTEM 20) during midyear. This is the current system.

The second year of operation confirmed some significant conjectures. Student interest was intense. The instruction received in the day-and-a-half workshop provided sufficient introductory background for the teachers to do a commendable job of using the computer system for a one month period. Indeed, the workshop was later reduced to one day. The impact on the teacher workload was more than trivial with students wanting early morning and late evening access to the computer. Relatively few problems were experienced with the computer system.

A used 8K PDP-8M EDUSYSTEM 20 was acquired on a lease/purchase arrangement, thus permitting 14 schools to participate each year. During the following four years the number of schools participating each year has been between 10 and 14. During those years when fewer than 14 schools participated some schools have elected to participate for two consecutive periods. These longer experiences indicate it is substantially more difficult for an inexperienced teacher to effectively use a computer in the classroom for two months than for one month. The classroom time for a one month period is used mostly to teach programming with some selected applications to the class topic. The second month typically requires more emphasis on applications and less on programming. The lack of teacher preparation and the length of time the computer is available to a school have not permitted teaching a computer science type of course. To provide more effective use of the computer, however, a number of teachers began to teach programming several days before the computer was delivered.

The staff responsible for the operation of the CLS does not develop curriculum materials. Instead, commercially available materials are provided in the Library of Materials. In an attempt to develop some materials specifically appropriate to schools in our state the CLS, in cooperation with Dickinson State College (North Dakota), and the State Department of Public Instruction sponsored a one-week summer workshop for North Dakota elementary/secondary teachers. These teachers prepared and documented materials for business, mathematics, science, and vocational education courses. The document, together with program tapes, are included in the Library of Materials. In addition to this workshop, the State Department of Public Instruction has funded some one-day programming workshops.

The CLS staff did, however, develop a manual that contains the essentials of BASIC, appropriate EDUSYSTEM 20 commands, and detailed step-by-step instructions on how to load EDUSYSTEM 20. A number of participating teachers indicated this manual contributed significantly to their success in using the computer system in their classrooms.

It was hoped that participating schools would recognize the value of computers in secondary education and acquire their own computer system. A survey of all schools participating in the CLS during the first 5 years of operation revealed considerable interest on the part of the teachers; only one school has acquired their own computer (a PDP-8 later replaced with a PDP 11/34). Another disappointment is that relatively fewer schools in the western part of the state participate. Although the eastern part of the state is the more populous, it is felt the reason for the difference in participation is largely because the University of North Dakota (UND) is located on the eastern boundary. This implies significant delivery/pick up charges for schools in western North Dakota. UND and the participating schools are illustrated in Figure 1. The participation of schools located in western North Dakota was the result of their teachers attending the week long workshop held at Dickinson State College in Dickinson, North Dakota.

In the past year the atmosphere for the use of computers in elementary/secondary education in North Dakota has changed drastically. The availability of microcomputers has provided a suitable computer system at an attainable cost. The manufacturer's advertising on radio and TV and in newspapers and magazines is viewed by students, teachers, and probably most importantly, administrators. Local stores, notably Radio Shack, are displaying micros, and students are pressuring the school system to purchase a micro.

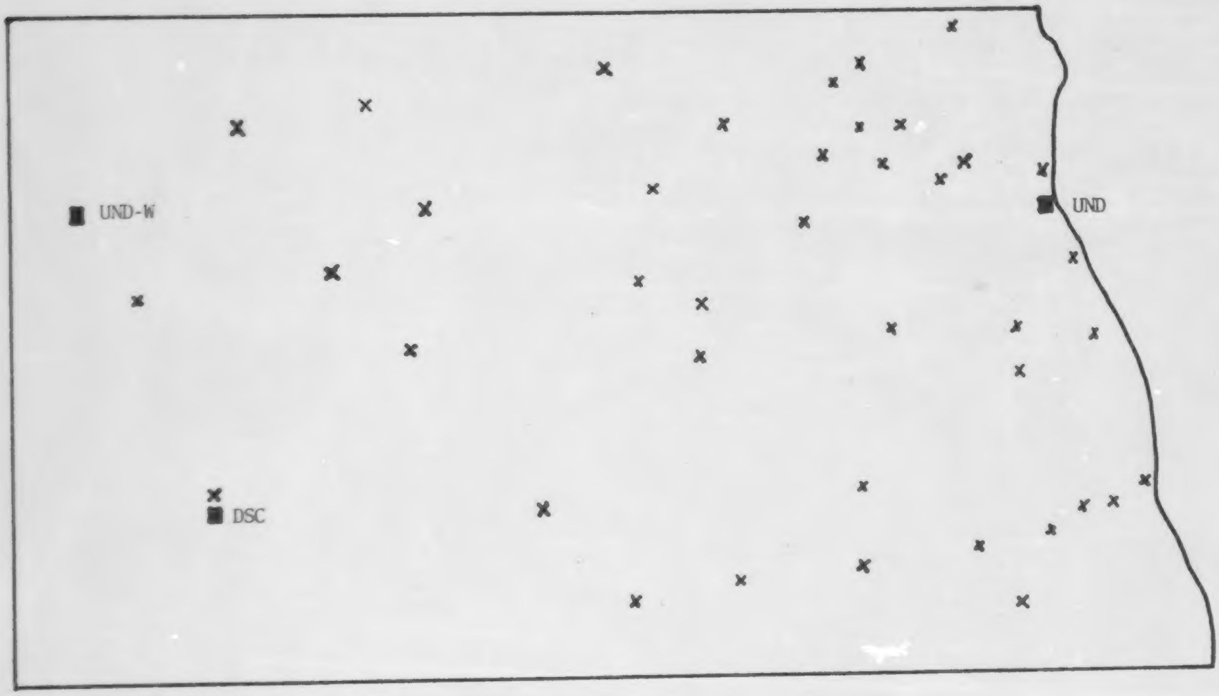
The CLS is responding to this growing enthusiasm by providing microcomputers (PET and TRS-80) for loan. Rental rates for a micro are considerably less than for a PDP-8. This in turn has reduced the PDP-8 rates. Expected rates effective next year are as follows:

	Rates	
Computer	First Month	Succeeding Months
PDP-8F and M	$ 105.	$ 80.
4K LEVEL II TRS-80	105.	80.
8K PET	115.	90.
16K LEVEL II TRS-80	130.	105.

The CLS is expecting to cooperate with the State Department of Public Instruction to develop a curriculum for a computer science course for our secondary schools. The CLS is also cooperating with the State Board for Vocational Education to provide computing support to business and vocational education teachers. The CLS, the State Department of Public Instruction and the State Board for Vocational Education are conducting one-day workshops in selected locations to introduce teachers to microcomputers and BASIC and to provide comparisons of various micros. The North Dakota Council of Teachers of Mathematics is initiating a software exchange service for microcomputer classroom programs.

Finally, to become more responsive to the needs of schools in western North Dakota, the CLS is establishing a branch operation located at the University of North Dakota-Williston Center (a two-year institution) located on the western boundary (indicated by UND-W in Figure 1).

In its sixth year of operation the CLS is enthusiastically looking forward to more years of interesting experiences.



X	Participating Schools
UND	University of North Dakota
UND-W	University of North Dakota-Williston Center
DSC	Dickinson State College

Figure 1

MANAGEMENT OF INSTRUCTIONAL COMPUTING:
Advice From Ten Pre-College Institutions

Beverly Hunter and Carol Hargan
Human Resources Research Organization

INTRODUCTION

The availability of inexpensive computer hardware is causing many schools and school districts to consider using computers for instructional purposes. Typically, neither the administrators nor the teachers in these schools has experience in the application of computers to learning and teaching. It is probable that each of these individual schools and districts will, over the next five or ten years, reinvent techniques and strategies for managing, developing, and using computer resources that have been reinvented over the past fifteen years. Nationally, the cost of this approach of introducing technology into the curriculum is enormous. Worse many schools may fail to introduce computers productively simply because teachers are not aware of factors that have been researched or discovered elsewhere.

This paper summarizes academic computing at ten precollege institutions, highlighting lessons learned through experience that might benefit other schools and districts.

THE CASE STUDY INSTITUTIONS

The institutions discussed in this paper were selected as the precollege case studies from 370 institutions participating in the project, "Exemplary Institutions in Academic Computing."[1,2,3] They range from a large city school district with over a half million students to a small private school of 950 students. The institutions, their enrollments and grade levels, are shown in Figure 1.

The institutions vary widely in the ways computers are used for learning and teaching and the resources that are expended for instructional computing. Some students use the computer perhaps an hour or two a year to receive vocational guidance; others use their computers a few minutes every day to receive drills in basic math and language skills. Other students use computers several hours every week to develop computer programs and solve complex problems. The proportion of students who access a computer some time during the school year ranges from about 4% to over 90% in these schools.

The case study institutions are not necessarily the most advanced, successful, or effective in instructional computing in the U.S. They do provide a broad representation of state-of-the-art applications and equipment found in educational practice. While no single institution provides a comprehensive model to be wholly adopted by other schools, many aspects of their instructional computing activities provide useful lessons and guidance to other educators.

OVER A DECADE OF INSTRUCTIONAL COMPUTING

The majority of our case study schools have been involved in instructional computing for over a decade. They have all found it to be a long, slow process to integrate computer use into the organization, the budget, the curriculum, and the classroom practices of teachers and students.

The case study schools and districts have all had some form of outside assistance in getting their computer projects underway. At least half of them received federal grants from the National Science Foundation and from the U.S. Office of Education. Others received help from computer manufacturers, regional and State computer networks, or universities.

Nearly all of the case study institutions were able to tie into timesharing

---

[1] Hunter, B., & Hargan, C. Instructional computing. . . ten case studies. Alexandria, Va.: Human Resources Research Organization, 1978.

[2] Academic computing directory (1st Ed.). Alexandria, Va.: Human Resources Research Organization, 1977.

[3] This research was supported by a National Science Foundation Grant No. SED 76-15399; Dr. Robert J. Seidel, Principal Investigator and Beverly Hunter Co-principal Investigator. Any opinions, findings, and conclusions expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation.

networks in the early stages of their computer activities. These included commercial timesharing networks, state and regional consortia, and university systems. Often these networks provided not only computer power, but also the instructional programs, teacher training, and technical expertise needed to get started. After an initial period of computing with this support, six of the schools and districts purchased or leased their own computers for instructional purposes.

Through many of the case histories we find a single, key individual who not only initiated computer-related activities, but also persevered despite administrative, financial, and technical obstacles.

ORGANIZATION OF COMPUTING

A wide variety of organizational and management arrangements are represented by the case institutions. As indicated in Figure 2, the computer facilities are controlled by the school district in most cases. In Dallas individual major projects control their own computers. George Washington High School owns several programmable calculators in addition to using the Denver Public School District's computer center. Lincoln High School uses a regional network, TIES. North Salem, Ridgewood, and Riverdale schools each have their own computers.

The computer facilities are shared between administrative and instructional applications in most of the cases. Ridgewood and A.I. duPont are the only two school districts that have computers dedicated to instruction. In the case of North Salem, the District's computer is used only for administration, and each school acquired its own instructional computers. The Coordinator of Instructional Computing at Ridgewood believes that, because of the diverse interests of the two groups, it is preferable to separate academic and administrative computing. She recommends both a separate staff and separate computing facilities for instructional use. At duPont, they believe that having the computer facility as an independent entity (not associated with the Math or Science departments) provides for smoother integration of computing into all disciplines and seems to help attract the less accomplished student.

Two-thirds of the schools and districts stressed that strong administrative support is necessary for a successful academic computing program. Chicago, for example, attributes some of the success of their Computer-Assisted Instruction program to the fact that it was jointly developed by educators, administrators, curriculum personnel, and data processing experts.

Two of the districts believe that although leadership from the administration is desirable, compelling individuals or schools to adopt a centrally developed computing program is not effective.

STAFFING

In at least half of the case study schools, the computer operations and programming staff consists primarily of high school students. Two of the case study institutions also use local college students as staff. In the case of Huntington Beach, their system has grown without needing to increase the staff of two half-time college interns. In addition, each of the case study institutions has one professional person who devotes full-time, or nearly full-time, to instructional computing activities. This person may be a Computer Teacher, a Coordinator of Instructional Computing, or a Computer Project Director. If a school has one main room where the computer terminals are kept, this room is usually supervised by teachers, student aides, or paraprofessional aides.

Riverdale's Computer Center Director states:

> One key individual who has the interest is the most important factor in initiating and promoting a good computer program. The individual must be given enough flexibility to experiment and develop the program without interference from the administration, yet maintain accountability. A good individual is more important than organization and funding at the beginning. Too much structure and red tape can kill any good intentions.

TEACHER TRAINING AND INCENTIVES

Computer literacy for the faculty has been an important aspect of the instructional computing activities of the case study institutions. The primary focus of the training is on computer applications that are useful to teachers in their classrooms.

In several cases, faculty have attended training programs at nearby universities. Graduate credit is obtained by participating in such courses.

Credit towards salary advances and promotions is sometimes provided as an incentive for faculty participating in training programs and in curriculum improvements. Other incentives include paid memberships to professional organizations such as AEDS, ACM, computer users groups, and the like; paid travel to professional conferences and workshops on instructional computing; stipends for computer activities; and the provision of technical support to faculty who want to improve their courses. Teacher training in computer applications has been partially supported by federal grants and contracts in at least four of the case study institutions.

Ridgewood tries to involve as many faculty as possible in the computer program and, if possible, provide salary incentives for participation (e.g., teaching an adult school class). They feel that ongoing, inservice training is a crucial component of any computer program.

In training teachers it is important to make them aware that they do not have to become technical experts in computing and programming in order to use computer applications in their classwork. Teachers should have an opportunity to have hands-on experience with computer applications in the subjects they teach, rather than simply receiving instruction in programming.

HARDWARE AND SOFTWARE

The computers used for instruction in the case study schools are listed in Figure 3. Several of the schools use microcomputers and programmable calculators. Others are in the process of evaluating microcomputers for particular applications. At Ridgewood, for example, curricula are being developed for elementary school use of the Commodore PET; microcomputers for use in physics are contemplated. Ridgewood is also integrating the use of programmable calculators into the junior high curriculum. At George Washington, a pilot study is underway to determine whether to upgrade the central computer or to replace timesharing terminals with microcomputers. North Salem plans to acquire an additional microcomputer each year, including one with color graphics. Riverdale is studying the application of microcomputers as nodes on a network; they also hope to use color graphics and animation.

At least half the institutions have found it advantageous to maintain access to a regional or commercial network for applications that require special facilities, such as a retrieval system for career and college information.

After trying to purchase a system to do everything, Mr. Jaquiss of North Salem now believes that many educational applications do not require a powerful central computer. In fact, many interesting projects can be accomplished with microcomputers. Even a $1000 computer system offers potential for hands-on use of a computer, including several advantages (e.g., screen graphics) not available in expensive time-sharing computer systems. The main disadvantage, particularly with several different models of microcomputers, is the lack of standardization among them.

The potential buyer must look carefully at the advantages and disadvantages of time-sharing minicomputer systems as compared to individual offerings in new microprocessor systems. Mr. Jaquiss of North Salem believes that criteria for evaluating computer systems have changed drastically in the years 1976-78. Although it is easy to calculate the cost per user-station per hour, it is very difficult to evaluate the intangibles of what the system will do for the user(s). Advertisements for computer products tend to be overly optimistic about good features and neglect to mention negative features.

Since the products on the market are changing so rapidly, Mr. Jaquiss hopes people planning for academic computing will be allowed to wait until they have the money to choose a computer. For example, in many schools, budget requests (including specifying a particular microprocessor and its price) for Fall 1979 were made in September 1978.

Carl Hauger of duPont recommends selecting computer systems that have the maximum flesibility to grow in terms of both hardware and software. In the instructional environment, flexibility is much more critical than in the business or administrative environment. He relies on other educational users as the best source of instructional software, particularly in light of limited educational budgets. The quantity, proximity, and variety of educational users with similar equipment should be paramount in the selection of a computer system.

Figure 4 shows the ratio of student users per terminal and student enrollment per terminal for the case study institutions. The wide variation in these ratios is a direct reflection of the kinds of computer use employed. In George Washington High School, for example, students work intensively with the computer in the computer math curriculum; hence a fairly low (40:1) ratio. At North Salem, by contrast, 700 students per year use the one terminal allocated to the guidance application. At Ridgewood, where nearly all students and many teachers are involved in computing, the availability of only ten ports has restricted the growth of instructional computing.

Most of the case study schools have found it advantageous to cluster the available terminals in a few locations rather than spreading them out among schools and departments. It is too difficult for a classroom teacher to integrate the use of one or two terminals into the activities of an entire class of students. Similarly, some have found the allocation of just ten minutes per student per day on a terminal to be logistically impractical, particularly where large numbers of students are involved.

COMPUTER-BASED CURRICULAR MATERIALS

Bruce Alcock of Riverdale believes that "finished instructional software" is one key to successful academic computing. Teachers, who in general do not write their own texts, will not produce their own instructional computer programs. He believes pressure should be placed on the government to start the curriculum development, and on manufacturers and publishers to produce quality packages that are flexible and easy to use. He also warns that one should not try to use the computer in too many curricular areas at once without adequate terminal time, hardware reliability, and training. He feels that it is important that novice users not have an initial bad experience with malfunctioning equipment.

COSTS AND FUNDING

Although many of the case study institutions benefitted from federal support in their early experience with computing, nearly all are now receiving their computing funds from local sources. In Chicago, the CAI activity (basic skills in reading and mathematics) is supported by Federal ESEA Title I funds, but the computer education program is locally supported. At Lincoln High School, the computing facilities and support are provided by the regional network (TIES), but the local district pays for the use of the network.

In many cases, this local support for instructional computing took many years to establish. Some of the schools and districts now enjoy the very positive support of the local governing boards. For example, in Huntington Beach a policy has been established that 1.2% of the school budget will be spent on computing.

While many school districts in the U.S. spend all of their computing dollars on administrative applications, one of the case study districts spends as much as 90% of the computing dollars on instructional computing. Dallas and Chicago, two large city school districts, allocate 24% and 28% of their total computing budgets to instructional as opposed to administrative applications.

The question, "What does it cost per student?" is very valid and important. The schools, however, do not maintain the kind of computer use and cost accounting data needed to provide unambiguous answers to the question. In very general terms, the case study schools range from less than $4.00 per student user per year, to over $160 per student user per year. They range from an average of $1.25 per enrolled student per year, to $29.00 per enrolled student per year for instructional computing equipment, staff, and materials. Glen Dysinger of Huntington Beach warns that once classroom computing is established, a district must anticipate expansion. Providing for the future income necessary to achieve growth must be an integral part of the program. Dallas found it advisable to initiate evaluation at the beginning of a project in order to have comparative documented evidence of a program's benefits as well as its costs for future budget and equipment allocations.

School District	1976-77 Enrollment	Grades	Year Computing Started	Estimated # Students Using Computers 1976-77	Estimated % Students Using Computers Annually
Chicago Public Schools, Illinois	513,000	K-12	1965	21,300	4%
Dallas Independent, Texas	140,000	K-12	1968	6,000	4%
* Alexis I. duPont, Wilmington, Delaware	3,000	K-12	1968	1,610	50%
* Huntington Beach Union California	20,000	9-12	1968	4,000	20%
School					
George Washington H.S., Denver, Colorado	2,000	10-12	1962	240	12%
* Lincoln, H.S., Bloomington, Minnesota	1,800	9-12	1965	900	50%
* North Salem H.S., Salem, Oregon	1,500	9-12	1968	750	50%
* Ridgewood H.S., Ridgewood, New Jersey	1,600	10-12	1966	1,500	94%
* Riverdale County School, Bronx, New York	950	K-12	1970	450	50%
Public Access					
* Lawrence Hall of Science, Berkeley, California		K-adult	1969	27,000	

* Institutions with computer literacy programs.

Figure 1. Profile of Precollege Case Study Institutions

School/District	Control of Computer Facilities Used for Instruction	Responsible for Instructional Computing	Reports to
Chicago Public School District	District	Bureau of Computer-Assisted Instruction	Dept. of Curriculum and Instruction Services
		Bureau of Computer Education	Dept. of Vocational and Career Education Program Development
Dallas District	District, plus Individual Projects	Individual Projects	Dept. of Research, Evaluation, and Information Systems
A.I. duPont District	District	Computer Project Director	Principal and Assistant Superintendent for Instruction
Huntington Beach Union High School District	District	Assistant Superintendent for Planning, Research, and Evaluation	Superintendent of Schools
George Washington High School	District, plus school-owned programmable calculators	Computer Math Teacher	District Supervisor of Mathematics Education and Curriculum Development
Lincoln High School	State Network TIES	Science Division Leader at Lincoln	Principal
North Salem High School	District	Computer Teacher	Principal
Ridgewood High School	District	Coordinator of Instructional Computing	Assistant Superintendent of Instruction, Principal
Riverdale Country School	School	Director of Computer Center	Headmaster of the School
Lawrence Hall of Science	LHS	Director of Computer Education and Operations	Director of LHS

Figure 2. Organization and Management of Computing

	Lawrence Hall of Science	George Washington High School	Ridgewood High School	Riverdale School	Lincoln High School	North Salem High School	Chicago Public Schools	Huntington Beach District	Dallas School District	A.I. duPont District
Altair									2	
Apple II	16									
Burroughs 6700									1	
CCC A16									2	*
CDC Cyber 73					*					
Data General Eclipse	3									
Data General Nova	1									
DEC PDP 11	*		1	1					1	1
DEC PDP 8						1				2
DEC Classic						1				
HP 2000			*		*				1	*
IBM System 3									1	
IBM 370							1	1	*	
Imsai						1			2	
PET	15		1			1				
Processor Technology SOL	3									
TRS-80									2	
Univac 1110							1			
Univac 1130		*								

*Access to

Figure 3. Main Computers Used for Instruction in Case Study Schools

Institution	Enrollment	# Terminals or Ports (Instructional Use)	Locations	Annual # Users	Student Users Per Terminal	Student Enrollment Per Terminal
Chicago Public Schools	513,000	921 CAI	58 Elem. Schools 2 High Schools	10,000/day	11/day	475
		163 Computer Education	4 Elem. Schools 64 High Schools	8,500	53	
		14 Guidance		800	57	
Dallas Schools	140,000	118	31 locations including all high schools	6000	46	1,077
A.I. duPont District	3,000	15	1 High School (computer room)	1,610	70	130
		8	6 Elem. Schools			
Huntington Beach H.S. District	20,000	60	6 High Schools–2 locations in each	4,000	66	333
George Washington H.S.	2,000	6+	Math Lab	240	40	333
Lincoln High School	1,800	7+	Open Science Lab Career Center Math Dept. Project Success	900	125	257
North Salem H.S.	1,500	1	Guidance Dept. (GIS)	700	700	300
		4	Computer Ed. Classroom	100	25	
Ridgewood H.S.	1,600	10+	Six locations in the H.S., including a Computer Resource Center	1,500	150	160
Riverdale School	950	15	Upper School terminal room; Middle School Math Lab; Lower School Math & Reading; 2 classrooms	450	30	63
Lawrence Hall of Science	NA	100	30 locations, many portable	30,000	300	NA

Figure 4. Terminals, Locations, and Student/Terminal Ratios

OZNAKI and BEYOND

Harvey A. Cohen,
La Trobe University,
Bundoora, Victoria, Australia 3083

OVERVIEW

OZNAKI, Polish for SYMBOLS, is the name of a project involving the use of contemporary computer technology to further mathematical education in primary and junior high schools. IN OZNAKI, microprocessor technology has been used to construct interactive models of mathematical systems. In these computer models, the student programs the activities of robots called NAKIs which have, in addition to a body state (vide the position, heading, and screen shape of the LOGO Turtle) a mind state having memory for both commands and the local environment. A rather novel scheme is used for passing numerical parameters to procedures without enforcing distinctions between formals and actuals. (i.e., no ", :, or the like indicator). The TINY languages in which the student programs these NAKIs are intended for implementation on exceedingly low cost controllers; for the moment personal computers based on 8080 and 6800 microprocessors are used in school studies, together with a minimum of specialised project hardware. Inherently the Project has a special interest in the microprocessor code production, update, and transport problem.

OZNAKI PROJECT: TEACHING AND EVALUATION

The Education Research and Development Committee of the Australian (Federal Government) Office of Education has supported the teaching and evaluation aspects of the OZNAKI systems PLUSMINUS, OZGRA, WHAM and CROC. Approximately seventy children have been directly involved with OZNAKI in state primary and secondary schools near La Trobe University campus.

The aim of this paper is provide an overview of the OZNAKI Project, to show how its objectives have been realised, and to discuss further and new directions for its development.

GOALS OF OZNAKI

The OZNAKI Project was initiated early in 1975 with the following avowed aims and objectives:

a) A committment to the analysis of the structure of mathematical understanding. That is, a concern with the cognitive processes (operations) involved in DOING mathematics, mathematics being rather broadly conceived.

b) To design computer controlled domains of robotic effects where the basic commands provide an embodiment of such OPERATIONS.

c) To extend the basic systems by means of a run-time macro facility so as to produce a family of clean robotic languages with a minimum of distracting computerese features. These languages to be similar in syntax to traditional algebra.

d) To design teaching modules that blend discovery with creativity. Such modules to have definite cognitive goals and involve a directed set of tasks somewhat akin to science laboratory units. In these set tasks, students by guided observation are lead to appreciate the significance of the operations that are modeled and the symbolic representation provided by key commands. These modules should also include mini-projects wherein children manipulate the symbolic operations (i.e., key in programs of their own design) to produce mathematical artifacts.

e) To use psychometric tests to evaluate the effectiveness of particular modules in achieving its stated objectives.

f) To demonstrate the possibilities of truly inexpensive microcomputers for personal use in Australian schools.

g) Insofar as purely computer concepts are to be presented, to develop stratagems of presentation that are purposeful and effective.

h) To investigate the practicality of developing the relevant computer software in a way that facilitates transport between different microcomputers.

THE PLUSMINUS

The young child who plays with the PLUSMINUS is confronted by two large keys, marked + and -, and a TV screen, on which a large 0 is displayed. If the large + key is struck, there is a sharp metallic click, a train appears on the screen, and the number displayed becomes a 1. Each further time the + key is struck, one further train appears, and the new total of trains is displayed. If the - key is struck, there is a loud click, the train at the end of the row disappears,and the number display is altered to show the new total. In place of trains, birds or dogs may be displayed.

In PLUSMINUS the number display together with the row of screen animals provide the same sort of embodiment of numbers as is given by the more traditional concrete materials, such as those of Zoltan Dienes. The PLUSMINUS associates a digit with a definite number of objects and with the number of times the + key has been struck in counting up to the screen total (or counting down to make the TV animals disappear.) Unlike rods, the PLUSMINUS also provides a concrete representation for the symbols + and - ; the actual OPERATIONS of addition and subtraction are modelled. Traditional materials can show only the before and after pictures for an arithmetical operation, and cannot model the operation itself.

A more detailed understanding of the significance of the PLUSMINUS can be derived from Gelman's studies of the development of early number concepts. To quote Gelman: "The cognitive processes by which people determine some quantity, such as the numerosity of a set of objects, are called ESTIMATORS. The cognitive processes by which people determine the consequences of transforming a quantity in various ways are called OPERATORS." As Gelman asserts, "operators are more central to a mature conception of number." It appears from various evidence, including the studies of Sinclair and Inhelder at Geneva, that the child's "number scheme is a central quantity scheme which facilitates the development of other quantity concepts."

The photo shows a 30 months old child using the PLUSMINUS. Note the large keyboard, as well as the microphone for speech input. (It is possible for the PLUSMINUS and other OZNAKI systems to accept speech input. We have experimented with a commercially available speech spectrum analyser. After a training session in which a child talks into a microphone attached to the microcomputer, particular words, up to 32 in current programs, of that speaker can be recognised. In the Speech PLUSMINUS the words "plus one" function exactly like the + keystroke, and the words "minus one" like the - key. The words "six" "dogs" will cause just 6 dogs to appear on the TV.)

ZONKY

ZONKY is a robot that crawls about the floor or table-top. Consistent with the general NAKI philosophy that a robot should have both a mind and a body, ZONKY has an electronic controller onboard that is to some extent indifferent to the form of ZONKY's body. Basically the controller will switch on two DC motors in various direction combinations, and will switch off each motor after a (programmable) number of revolutions. (The controller counts the spikes on the line due to the sectoring of a commutator.) The original ZONKY had as body a Sherman Tank, made by the Takimaya plastic toy company of Japan. ZONKY II had a circular plan, and with the pen holder mounted precisely in its geometric centre, had improved accuracy. However students definitely preferred the more toy-like original ZONKY, so that in the radio controlled ZONKY III once more a clackedy model tank was used adapted. On any ZONKY are marked the FORWARD direction and the directions for RIGHT and LEFT turns (on the spot) which take place about a central point where a felt tipped pen is mounted. This pen, if inserted, marks out the robot's trail.

The basic movement commands for ZONKY are F for forward, B for backwards, R for right, and L for left. While performing these movements the automaton can also be ordered to honk, H, glow, G (switching on its lights). A string of commands - - like an English command - - must be terminated by an exclamation mark ! (There is a version of the command language for pre-schoolers where the ! is not required, and only one command at a time can be performed.) To illustrate, following the command 3GF 2HR ! the robot makes 3 steps forward with its lights flashing, then makes 2 turns right (clockwise) on the spot whilst honking its horn. In between each forward step or unit turn there is a one second delay. On the TV screen the commands punched in by the student are visible; also displayed are the actual control words that are directed at ZONKY. Thus for the students commands,

3HF 2GR

a further embodiment of number is supplied through the display on the screen of the ZONKY language words.

```
aaaaaaaaaa     aaaaaaaaaa     aaaaaaaaaa
gggggggggg gggggggggg
```

Note that corresponding to a unit motor command, there are precisely ten lower case control characters displayed. This facilitates the introduction of decimals.

WHAM

In WHAM the student controls the happenings on a TV screen with a typewriter style keyboard. Most of screen is taken up with a square in which a strange creature called a NAKI lives. This creature obeys the same basic commands as the OZNAKI robot ZONKY, namely FORWARD, BACK, RIGHT, and LEFT. These commands are represented by the typewriter keys F, B, R, and L. However the NAKI only turns ninety degrees, e.g. from being

V

facing down (or South) on the TV screen, to the state

<

after the command R, now the NAKI is heading West. As the NAKI roams its territory it leaves a trail of asterisks, so forming a design.

So far we have described the body of the WHAM NAKI, and the movements of this body. The NAKI also has a mind visible on the screen but strangely detached from the body. In the NAKI's mind is the last command it obeyed, together with the total of what it has learned. Of course, on starting a session of WHAM the NAKI knows nothing. But each time the Z command is used to make it remember some list of commands by name, that list and its name are displayed in the mind. In the copy of the WHAM screen below the NAKI's body has wandered around the area fenced off with + signs. To the right is the NAKI's mind.

In WHAM the command P clears the NAKI enclosure (bounded by + signs) and returns the NAKI to its HOME at the centre of the enclosure. So there is great freedom for the child to experiment while gaining experience with the basic movement commands. To draw the logo design shown above, a child was asked to direct the NAKI from HOME to the top right hand corner square by a squiggly path, ending up heading east. (Overall, along this path, the NAKI has turned through 90 degrees). The child was then asked to teach the NAKI how to draw such a path by punching in Z, then V (the name of what is to be remembered) and then the commands that the NAKI followed in sequence. Thus the child had to work backwards along the marked path on the screen to fill out the memory list. This list is, of course, a symbolic representation of the path, SYMBOLS being what OZNAKI is about. Having checked that this V command was the one intended, clearing the screen and trying again if it wasn't, the child was ready for a direct suggestion from his teacher. The command J returns the NAKI to HOME, without changing the direction in which the NAKI is facing. So by punching in ZW V J ! the NAKI learns how to draw an entire arm of the logo, followed by a jump home. Having cleared the screen with the P command, the NAKI could then trace out the entire logo on the command 4W! Note how much issues of planning and problem decomposition enter into drawing such a logo.

But this is only part of the WHAM story. Also demarcated on the screen is a little area containing a number. This number is in fact the display of an electronic calculator. Unlike the defined macros displayed on the TV, this number is not actually in the WHAM NAKIs mind, but in the mind of the WIZARD (of OZ , naturally) who is the overlord of the system. (In a multi-NAKI environment this number, together with a few others, can be used by ALL NAKIs.) The name of this number is A. One can enter numbers in the display and perform arithmetic, just as in any reverse Polish calculator. Thus 2+ adds 2 to the number displayed in the accumulator, while 3- subtracts 3 from

```
+ + + + + + + + + + + + + +
+  *                      +
+  *                 * * *+  V=2F R 4F L 3F R 2F
+  * * * *           *    +
+        *           *    +  W=V J
+        *   * * * * *    +
+        *   *            +  X=
+        * * ↑ * *        +
+            *   *        +  Y=
+    * * * * *   *        +               + + + + +
+    *           *        +               +       +
+    *           * * * *  +               +     0 +
+* * *                 *  +               +       +
+                      *  +               + + + + +
+ + + + + + + + + + + + + +    4W!
```

```
+ + + + + + + + + + + + + +
+                         +
+                         +  V=
+    * * * * * * * * * >  +
+    *                    +  W=
+    *   * * * * * *      +
+    *   *         *      +  X=AF R 1+
+    *   *   * *   *      +
+    *   *     *   *      +  Y=
+    *   * * * *   *      +               + + + + +
+    *             *      +               +       +
+    * * * * * * * *      +               +    10 +
+                         +               +       +
+                         +               + + + + +
+ + + + + + + + + + + + + +    10X!
```

the number display. Combining this calculator arithmetic with the NAKI movement commands yields a numerical geometry which includes not only squares and rectangles, but also several forms of spiral. In the spiral shown with the TV screen for WHAM on the bottom of the page previous, the student had cleared the game area with the P command, and reset the A accumulator with the command A-, elsewise using the command 0E (E for Enter). On an X command, the NAKI moves forward A steps, turns right, then A is increased by 1. The spiral depicted was drawn as shown on the command 10X!

The similar spiral in the line-graphics of LOGO is termed by Papert a squiral. It is worth comparing the clean traditional algebraic formalism used in WHAM to draw a squiral with the more computerese recipe of LOGO.

OZ/TINMAN

The language in which children program the walking automata ZONKY was originally called OZ. With the introduction of a front panel display of the NAKI mind (remembered command macros) together with primary and secondary accumulators A and B, this language was tuned up to become TINMAN. (Perhaps in a later version TINMAN's heart will also be displayed, and the robot will be truly complete.)

```
         *
        ***
       *****
      *******
     *********
    ***********
   *************
  ***************
 *****************
*******************
         *
         *
    ***********
     *********
      *******
```

In TINMAN the messages from the WIZARD to ZONKY appear on the left hand half of the TV screen, and scroll upwards. Using / simple patterns of characters can be formed on the screen, messages can be printed by macros, and built-in commands such as F can be duplicated since all type-out goes to ZONKY. The incorporation into macros involving / of the simple number handling capabilities of OZ/TINMAN can give interesting patterns, such as the XMAS tree design above.

ZONKY is a Johnny One-Note which can be programmed to sing. There are just two commands for this purpose, T to toot for unit time duration, P to pause - - do nothing for unit time duration. Built into the command language is the command J, a composite macro whose contents students can see, though not modify.

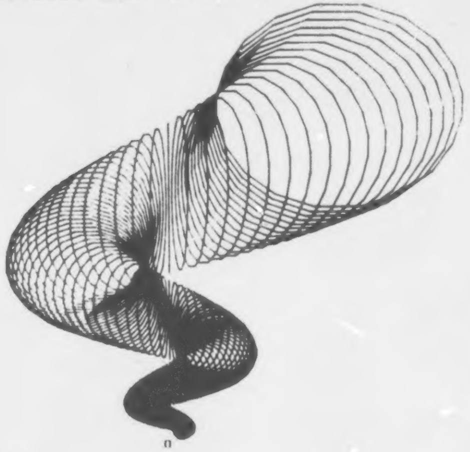
```
J=3T 2P 3T 2P 5T 4P /J /I /N /G /L /E / /B
/E /L /L /S / /! /
```

The command J causes ZONKY to give quite a credible rendering of the opening bars of "Jingle Bells", followed by the appearance of the actual words on the screen, as specified by the printing primitive "/".

OZGRA

OZGRA, also called OZ-Graphics, is a line-graphics language closely related to LOGO. In fact OZGRA has some picture manipulation primitives not available in LOGO. Our implementation uses a Tektronix GT1024 graphics terminal for display. A detailed description of OZGRA is given in the article by H. A. Cohen and D. G. Green (1977).

OZGRA has been tailored to the processing capabilities of a microcomputer with limited memory. OZGRA requires a fine graphics screen which costs at the moment of the order of two thousand dollars. For this reason OZGRA has been excluded from our school trials.



LIFE

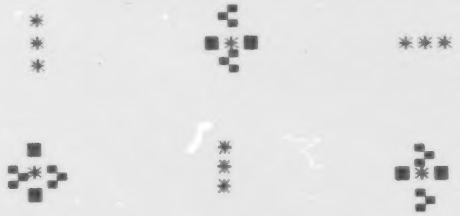
The game (?) of Life was discovered in 1969 by the Cambridge mathematician Conway in his studies of cellular automata. In school trials of OZNAKI, a new computer version of Life has been used, in which

Conway's Life provides growth and life to drawings composed by the user.

In this microcomputer Life, the user first guides about the TV screen a NAKI which obeys the commands, N for North, S for South, E for East, and W for West, each command moves the Naki just one step about the screen composed of 16 rows of 64 cells. On the command L - - for Life - - the NAKI puts life into the cell currently occupied, the cell being marked with an asterisk. Similarly, on the command K, a cell - if live - becomes dead, so that the asterisk disappears.

Each cell on the screen has eight immediate neighbours. If a cell has two live neighbours, it will be unchanged in the next generation. If any cell has three live neighbours then in the next generation that cell will be live. For other than two or three live neighbours, any cell will be dead in the next generation. That is, these simple creatures can neither abide loneliness nor overcrowding.

In Wizard Box Life, the student can single step from one generation to the next, or set the process going so that each new generation is displayed for just one minute. In addition, on the command H, a half-way state is indicated, whereby live cells that are doomed to die on the birth of the neet generation are indicated by a trident graphic character, while dead cells destined for life in that generation are marked with a rectangular mark. An example is provided by the progress of a set of traffic blinkers.



In working with school children a number of simple patterns are supplied or suggested that give easily understandable results when the Life process gives birth to a new generation.

EVALUATION PROGRAMME

How does one evaluate the educational robotics developed by the OZNAKI Project? Essentially there are two (intersecting) forms of evaluation. Typically educational systems (broadly conceived) are evaluated on a behaviorist basis: the student is conceived as some sort of black box, of unknown and unknowable inner structure, and the difference in behavior (equated to class marks and the like) is measured before and after exposure to the system.

In contrast, the OZNAKI Project is concerned with the difficult task of defining cognitive goals. To specify a cognitive objective, we must state a set of changes we want to bring about in the student's cognitive processes. Thus we are inherently concerned with the inner structuring of knowledge - - where the inner is the contents of the behaviorist's black box.

It is in this general philosophic framework that studies of the OZNAKI Project were conducted in 1977. In the 1977 evaluation study we sought to examine enhancement of spatial abilities. We focussed on projection - - the ability to see with the mind's eye from elsewhere - - this having been demonstrated by Piaget as a major process in spatial thinking.

Dr. David Green and this writer designed an introductory course in OZNAKI with a pronounced empahasis on projection. In this course of just eight lessons, the first lesson involved ZONKY, our real robot, then WHAM, and finally LIFE. This module provides an introduction to the algorithmic aspects of OZNAKI, including WHAM calculator maths, some geometry, and music.

The evaluation study carried out in Spring 1977 had the following phases:

(a)Pretesting of all students involved.

(b)Selection of Experimental and Control Groups.

(c)Teaching the Projection Module to the E Group.

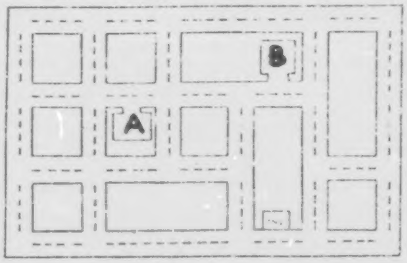
(d)Post-testing of both E and C Groups.

The students participating in the study were aged 8 to 13. They included both primary and secondary students. Similar, but distinct, experiments were carried out at the high school and primary schools. Primary students were tested in Piagetian interviews, whereas the secondary students used multi-choice questionnaires.

A detailed account of the 1977 evaluation study of the Projection Module is given elsewhere. In sum, the results derived from this study were dramatic. Following a course of just eight lessons in OZNAKI, children improved remarkably in spatial skills.In certain of the Piagetian interviews used in the evaluation, the student is asked to guide a model car about the streets of a model town, keeping the car on the correct side of the dual carriageways. In the "Left-Handed Route" problem the subject was asked to guide the toy car from car-park A to car-park B, as marked in the map below.

Drive
from
A to
B

Make
left
turns
only



In the pretest only 4 members of the experimental and also 4 members of the control group of primary students could find a route satisfying the conditions given. On the post-test, of the experimental group, 21 (out of 27) students found a solution, whereas in the control group only 3 found correct solutions. Other improvements were so large and generally consistent that despite the small sample sizes involved (22 or 27 in each of the groups), statistical analysis confirms that highly significant changes were achieved in the children's projective skills.

That such good results can be achieved with such short exposure to OZNAKI highlights the importance of the Projection Module as a paradigm for the development of other courses of instruction involving computer embodiment of mathematics. In all phases, the hardware and software, and the lesson structure, the module was designed with definite cognitive goals in mind. Having definite objectives, it was possible to measure effectiveness in reaching these objectives.

BEYOND OZNAKI

The OZNAKI Project is now in its fourth year of operation. The initial hardware and software goals have been reached with the systems described above. At this date there is only a very modest amount of teaching experience in schools. The 1977 Evaluation Trial supplied a paradigm for evaluation, and more extensive work in the same spirit is scheduled for 1978. The emerging question is what lies beyond the core of OZNAKI as described above. A full answer is not attempted here, rather some departures from the original framework are chronicled.

WIZ77

WIZ77 represents a first attempt to go beyond the confines of four user-defined macros. Just as in WHAM, on the right hand side of the screen are displayed the user's command line and user-defined macros. There is room on the screen for 14 such macros. On the top right are displayed the contents of two accumulators, A and B. The arithmetic operation + adds one to A, while - causes one to be subtracted from A. Numbers are entered into B by a SWAP command that swaps the contents of A and B stores. Thus two numeric parameters are available to user macros without complicating the simple calculator arithmetic introduced first in WHAM. The really nice feature of two accumulators is that I was able to implement conditionals that compared A with B. For instance, the symbol "=" has value 1 if A and B are equal, zero otherwise. By using this symbol before a bracket, the bracket contents are executed if A=B. Incidentally, the incorporation of brackets is very natural in macro languages. In WIZ77 a bracketed expression may be closed either by a right-hand bracket or by the "DO-IT" symbol. The original implementation runs a WHAM NAKI that can turn in units of 45 degrees. This NAKI can sense the contents of the cell immediately ahead and hence steer around obstacles.

WIZ77 is seen as a universal TINY robotics language. In future versions of WIZ77 the user will be able to name a particular NAKI so that, for instance, having written a series of macros for drawing stars the student might simply invent the name STARMAN. Then on storing away STARMAN, using the mass storage facilities of the microcomputer, the student could in future sessions simply summon STARMAN to reappear.

COMPUTER ART

It is clear that in any OZNAKI-like system students are using mathematics: can what they do really be art? If the microcomputer graphics medium was used by children to produce stereotypes of trucks, houses, stick figures, and the like the students might be kept busy, but this activity could hardly be called art. (In the same spirit concerned art teachers have condemned the use by children of ditto master outline drawings that children fill-in with coloured crayons.)

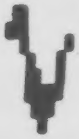
In order to expand the gambit of OZNAKI to include art, it became essential to have an understanding of the evolution of the artistic capacities of children. Studies of the artistic development of children have for historical reasons focussed on representational art. Recently, Drora Booth of La Trobe University has completed a study of the development of painting skills in preschoolers attending a child minding centre fully involved in her experiment. The unique feature of this experiment was the protection of the children from the usual pressures to produce representational art. All paintings (which were never called by the name "pictures") were kept at the center. To reduce the pressures on the children to produce representational art, the objective of the experiment was explained to their parents. The children had regular sessions for what was called painting (the medium being not commonly used at home.) Over a year, more than two thousand paintings were produced which were then analysed to show a clear-cut developmental pattern. Booth's work showed that children spontaneously discover ideas of pattern, and in progressing through stages make more and more elaborate use of the mathematical (!) ideas of repetition of a basic motif on a rectangular lattice, with reflection and variants of glide symmetry.

Although pattern drawing has been almost totally neglected in traditional school education due to historical reasons (as discussed by Booth in her thesis (1978)), her study demonstrated that it was a natural element in children's art, with perhaps not-so-suprising links with spatial ideas in mathematics. There are signal prospects for "Education through Art" (as espoused by Read) by linking a drawing system with a mathematical formalism. One must provide the child with a facility for repetition of motifs, combination, translation, reflection, and rotation operations. A graphics language for microcomputers with such capabilities is TV Sketch.

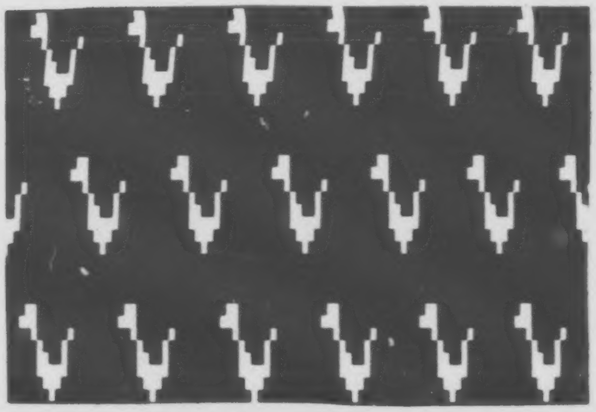
TV SKETCH

TV Sketch is a block graphics language, rather different to the line (or streak) drawing languages of more conventional computer graphics. TV Sketch was originally conceived as a vehicle for teachers to use to write their own extended version of PLUSMINUS. It does seem naturally related to children's spontaneous pattern making, and is being adapted for use by children.

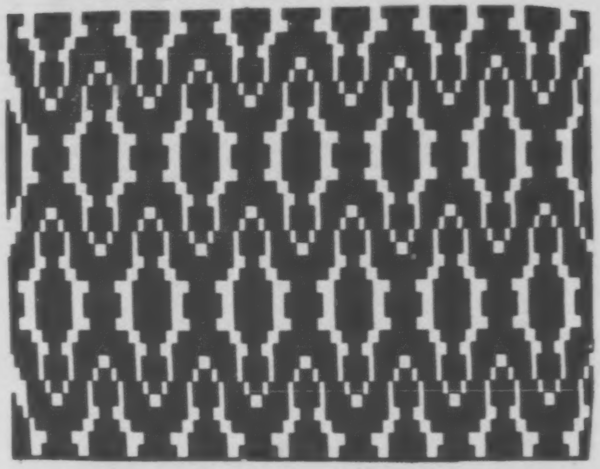
In TV Sketch the motif is a primitive construct, owned by a NAKI, whose existence simplifies the repetition and combination of a motif over a lattice. Reflection and colour inversion are basic primitives of TV Sketch.



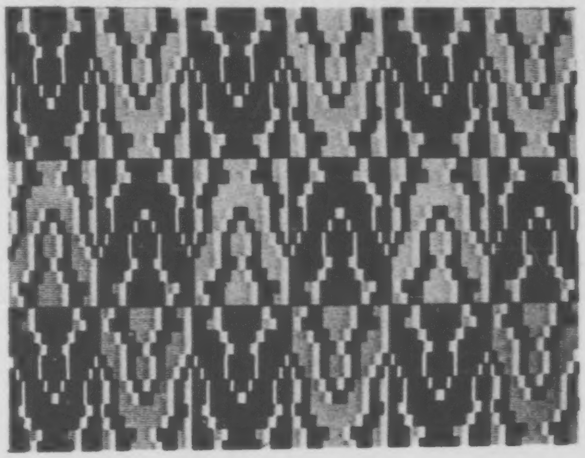
To explain the general idea, suppose the user creates the NAKI motif to the left of this text. (If you insist on a representational interpretation this might be a bird.) Suppose now the NAKI drags this motif around the screen and dumps the motif in a regular manner.



Reflecting this pattern twice yields



Reversing regularly about the pattern gives



SYMBOL GRAPHICS IN SPECIAL EDUCATION

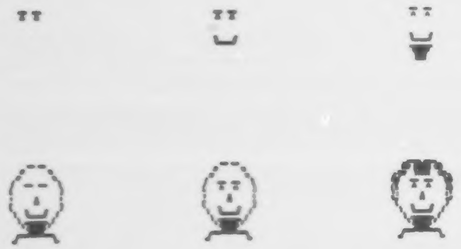
The scope of OZNAKI has been the fostering of the intellectual development of normal children. Recently the author and Ms. Robyn Hannan have been applying the spirit of OZNAKI to the needs of special education.

As demonstrated above, OZNAKI has been concerned with the construction of domains in which keyboard commands effect transformations that model some cognitive operation. Even though what is being modeled might be a relation, in OZNAKI it becomes a command (transitive verb!). The base level of symbol graphics is to use a keystroke to stand for a thing (noun) so that the symbol on the key is more precisely the command that the thing denoted appear, disappear, or move about.

The children we are now working with have considerable difficulty acquiring reading skills, so that it is not helpful to use alphabetical letters as key-symbols. We have chosen to use the pictorgraphic language of BLISSYMBOLS.

The first system developed is called HANGMAN (because of its resemblence to the well-known children's game.) Computer power is supplied by an OZNAKI microcomputer (Wizard's Box). When the game commences, on the screen the child sees an outline face. On the twelve keys of a giant keyboard are the Blissymbols for eyes, mouth, hair, and nose. By pressing the appropriate large key, the corresponding picture element appears. If the face feature is already displayed, pressing the corresponding key causes it to disappear. The child in the photo below is one of a group of Down's Syndrome children at the Moorabbin Day Training Centre, Melbourne, Australia.

The film strip below shows (in colour reverse) the entire face, and various partial faces the student can produce enroute.



The PLUSMINUS is relevant in this enviroment for presenting the most basic ideas of numbers and number operations. The children we deal with lack the more basic spatial skills. As the TV screen is near vertical, UP/DOWN is easily modeled, while IN/OUT can be made key commands to move displa~ed objects in and out of given areas. HANGMAN was implemented in TV SKETCH, which will be appropriate for constructing these spatial microworlds.

BIBLIOGRAPHY OF OZNAKI LITERATURE

OZNAKI - General
* H. A. Cohen, "Micro-Computers for Mini-Mathematicians", in L. Murray (Editor), "Meaningful Mathematics", Published by the Mathematics Association of Victoria, 1977, pp 391-7.
* H. A. Cohen, "OZNAKI: A New Medium for Mathematicians", in D. Williams (Editor), "Learning and Applying Mathematics", Published by the Australian Association of Mathematics Teachers, 1978, pp 274-83.
* H. A. Cohen, "Expanding the Child's Concept of Number, Space and Operation", in M. Poole (Editor), "From Creativity to Curriculum", to be published.

ORIGINAL DESCRIPTION of OZ/TINMAN
* H. A. Cohen, "The OZNAKI Robotics Language OZ", Proceedings of the 7th Australian Computer Conference, Vol. 1, 1976, pp 128-143.

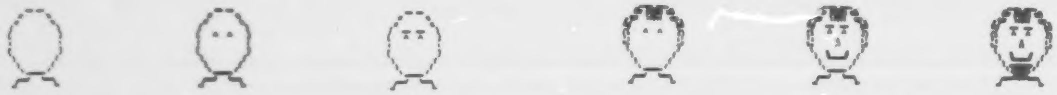
ORIGINAL DESCRIPTION OF PLUSMINUS
* H. A. Cohen, "OZNAKI", in "Set Two", published by Mathematics Association of Victoria, Melbourne, 1976, pp 36-9.

EVALUATION
* H. A. Cohen and D. G. Green, "Evaluation of the Cognitive Goals of OZNAKI: Enhancement of Spatial Projective Abilities", in "ACM Topics in Instructional Computing" (a special publication of ACM Sigcue), A. M. Wildberger and R. G. Montanelli, eds., ACM, New York, 1978, pp 69-90.

OZNAKI version of "LIFE"
* H. A. Cohen, "The OZNAKI "Life"", Dr Dobb's Journal of Computer Calisthenics and Orthodontia, Published by People's Computer Company, Menlo Park, California, Vol. 3, No. 4, (1978) pp 10-11.

OZGRA
* H. A. Cohen and D. G. Green, "Teaching Mathematics with OZ-Graphics", Peoples Computers, 1977, vol 6 (3), pp 52-58.

CHILDREN'S DRAWING
* D. Booth, "Pattern painting by the young child: the roots of aesthetic development", Australian J. of Education, 20 (1976) pp 110-112.
* D. Booth, "Child Art and pattern painting", Australian J. of Early Childhood, 2(1977) pp 14-21.
* D. Booth, "Curriculum and the Young Child's Spontaneous Geometric Pattern Making", in M. Poole (Editor), "From Creativity to Curriculum", to be published.
* J. Goodnow, "Children's Drawing", Fontana, 1977, pp 40-43.



MICROPROCESSOR CODE PRODUCTION, TRANSPORT, PORTABILITY, STRUCTURING

* H. A. Cohen and R. S. Francis, "Programming Constructs for Microprocessors and Bit-Slice Processors", Institute Radio and Electrical Engineers IREECON Conference Digest, Melbourne August 1977, pp 119-121.
* H. A. Cohen and R. S. Francis, "HELP for Microprocessor Software Development", IEEE Compcon Fall Conference Digest, Washington, September 1977, pp 196-200.
* H. A. .Cohen, "Microprocessor Software Development Using Macroprocessors", in Proceedings of Sth Conference of the Australian Computer Society, Canberra August 1978, Vol. 1, pp 148-164.
* H. A. Cohen and R. S. Francis, "Macro-Assemblers and Macro-Based Languages in Microprocessor Software Development", IEEE Computer, Vol 12, No 2, Feb 1979, pp 53-64.

GENERAL BIBLIOGRAPHY

EDUCATION AND EVALUATION

* J. B. Carroll, "Psychometric tests as cognitive tasks: A new "structure of intellect" in L. B. Resnick (Ed.) "The nature of intelligence", Lawrence Erlbaum, Hillsdale Ill., 1974 pp 74-116.

EXPERIMENTAL DESIGN, STATISTICAL ANALYSIS

* T. W. Anderson, "Multivariate Statistical Analysis", New York, Wiley and Sons,1958.
* L. J. Cronbach, G. C. Oleser, H. Nanda, and N. Rajaratnam, "The dependability of behavioural measurements: Theory of generalizability for scores and profiles", Wiley, New York, 1972.
* L. J. Cronbach and L. Furby, "How should we measure "change" - or should we?" Psychological Bulletin, 74(1) 1970 pp 68-80.
* N. H. Nie, C. H. Hull, J. G. Jenkins, K. Steinbrenner, and D. H. Bent, "Statistical Package for the Social Sciences (SPSS)" New York, McGraw Hill, 1975.

SPATIAL and MATHEMATICAL ABILITIES

* V. A. Krutetskii, "The Psychology of Mathematical Abilities in Schoolchildren, University of Chicago Press, 1976.
* I. McF. Smith, "Spatial ability: Its educational and social significance" University of London Press, London, 1964.

BLISSYMBOLS

* C. K. Bliss,"Semantography/Blissymbolics" Semantography Publications, Sydney, 2nd Ed. 1965.

NUMBER CONCEPTS

* Rochel Gelman, "The Nature and Devlopment of Early Number Concepts" in H. W. Reese (Ed.), "Advances in Child Development and Behavior" Vol 7 ,New York, Academic Press 1972
* Rochel Gelman and Marsha F. Tucker, "Further Investigations of the Young Childs Conception of Number" in Child Development, Vol 46, p167, 1975.

COGNITIVE GOALS IN EDUCATION

* L. B. Resnick, "Task analysis in instructional design: Some cases from Mathematics", in D. Klahr (Ed.), "Cognition and Instruction", Lawrence Erlbaum, Hillsdale N.J., 1977 pp 51-80.

SPATIAL PERCEPTION AND GEOMETRY

* P. G. Gould and R. White, "Mental Maps", Pelican Books, Middlesex, England, 1974.
* J. Piaget and B. Inhelder, "The Child's Conception of Space", Available in several editions.
* J. Piaget, B. Inhelder and A. Szeminska "The Childs Conception of Geometry" Basic Books, New York, 1960.

CONSERVATION CONCEPTS

* H. A. Cohen, "The Art of Snaring Dragons", M.I.T. Artificial Intelligence Laboratory Memo 338, May 1975.

MULTIPLE EMBODIMENT

* Z. P. Dienes, "Building up mathematics", Hutchinson, London, 1958.

LOGO

* Jeanne Bamberger, "What's in a Tune", Logo Memo No. 13, M.I.T. Artificial Intelligence Laboratory, July, 1974.
* Seymour Papert, "Uses of Technology to Enhance Education", LOGO Memo No. 8, June 1973, M.I.T. Artificial Intelligence Laboratory.
* Seymour Papert, "Teaching Children to be Mathematicians VS Teaching About Mathematics", LOGO Memo No. 4, July 1971, M.I.T. Artificial Intelligence Laboratory.

SMALLTALK

* Deidre Goldberg and Alan Kay, "Personal Dynamic Media", Xerox Palo Alto Research Center Report, 1976.
* Alan Kay, "Smalltalk", in Scientific American, September 1977.

ACKNOWLEDGEMENTS

The teaching and evaluation studies described here were supported by the Education Research and Development Committee (E.R.D.C.) of the Australian Office of Education.

SIMULATING THE GREAT DEPRESSION IN INTRODUCTORY MACROECONOMICS

Robert Schenk*
Saint Joseph's College
Rensselaer, Indiana 47978

One goal that many teachers of undergraduate economics have is to help students master certain theories or models. Most students will eventually find careers in business or government where their working environment will be shaped a great deal by economic forces and economic theories. Teaching students to think in terms of economic models will help them cope with that environment.

Many instructors have found that use of computer simulation helps students understand macroeconomic models. To operate successfully in a world of a simulation, a student must go beyond mere memorization of the material to application and analysis. Further, many instructors have claimed that use of computer simulation increases student interest in subject material.

Most existing macroeconomic simulations are simulations of hypothetical or fictional economies. A problem with these simulations is that they ignore and obscure the problem of validation. Many students, and even some instructors, may accept these simulations as truth rather than as conjecture about how the world works. For example, one author has written that his simulation illustrates to the students "that the economy really does operate the way course lectures and readings have said it does. The mutual interaction of mixing the real world with the textbook world has, experience has shown, strengthened the understanding of each."[1] Given that only two paragraphs earlier he had justified his use of imaginary data, this is truly a remarkable conclusion.

Every theory is conjecture about how the world works. A computer simulation is an application of the theory. "A major difficulty with any computer simulation project is determining the validity of the simulation. That is, does the model as operated in the simulation really reflect the real-world parallel?"[2] None of the hypothetical-data simulations permits one to deal with this problem.

Existing simulations using real-world data tend to be complex. Complex simulations are subject to Bonini's paradox: a purpose of the simulation is to achieve understanding, but as it becomes increasingly complex, the simulation becomes no more easy to understand than the real-world process.[3]

An Alternative Approach

Most instructors emphasize a few simple models in principles of economics. This emphasis on simple models can be justified only if these models roughly approximate the real-world system under study. Certainly a model should not be taught in a form so simplified that it would lead to misunderstanding. Furthermore, if a model does tell us something about the world, we should use it to explain real situations. Thus if we are willing to draw policy conclusions from a simple Keynesian model, as many textbooks do, we should be willing to use that model with real-world data to explain past economic performance. These considerations have led me to take a simple Keynesian model and a variant of the Quantity Theory of Money, two commonly used models in introductory courses, and use them to interpret the most dramatic macroeconomic episode in American history, the era of the Great Depression. This episode is especially interesting because both the Keynesian revolution and the monetarist counterrevolution emerged from attempts to explain it. Keynesian economics was a sensational success because it offered an explanation of this period which, it claimed, was inexplicable when viewed from classical models. The modern rise of monetarism dates from 1963 when Milton Friedman and Anna Schwartz demonstrated that the Depression was explainable in terms of changes in the money stock.[4]

The sample outputs in the Appendix provide descriptions of the models on which the simulations are based and show examples of student-machine interaction. Several points deserve comment. The multiplier in the Keynesian simulation is three because this integer gave better results than others and use of an integer keeps the model simple. In the Quantity Theory simulation velocity is not a constant but a variable determined endogenously, falling when money stock falls. Though introductory textbooks usually do not present the model in this form, few if any Quantity Theorists have maintained that velocity is constant in the short run.[5]

The mechanics of these simulations are similar to those of other macroeconomic simulations. The student relives history as a policymaker, controlling either money stock or government spending and taxes with the goal of keeping the economy growing smoothly with neither inflation nor unemployment. After each policy decision the computer computes a simulated GNP and evaluates the student's expertise by calculating rates of inflation and unemployment. The unemployment rate depends on the gap between potential and simulated GNP and the rate of change in simulated GNP. (Thus it is possible to get unemployment and inflation simultaneously.) These rates are printed only when they are at a level indicating that a student is doing a poor job as a policymaker;

other comments may also be added. At the end of the simulation, the computer evaluates a student's overall performance and gives some hints on how to improve if they seem needed.

Advantages of This Approach

Simple, real-data simulations seem to teach the mechanics of manipulating policy as well as simple, fictional-data simulations. Students must understand the multiplier principle, for example, to successfully manage Keynesian policy in both types of simulation. There are also several advantages to the use of these simple, real-data simulations.

One advantage is that they indicate to students that economics may have important things to tell us about history and that historical episodes shape our theory. Many instructors try to show the contemporary relevance of macroeconomics by using current economic events to highlight or illustrate important points. However there are often past events which are as interesting or important. The decade of the Great Depression, for example, is the key macroeconomic event in history because the central problem in macroeconomics has been to explain how it was possible in a market economy for the mass unemployment of this period to develop and then persist.

A second advantage is that they present realistically the present state of macroeconomics — there is not consensus among macroeconomists but rather deep division. These two simulations help explain why. Given their simple structure, both do a remarkable job tracking GNP for the twelve years. It should be apparent to the student that sometimes it is not easy to empirically decide between alternative theories, and as a result two competing theories may coexist for long periods of time.

A third advantage is that this approach raises new questions. One is the question of validation: which model is a better approximation to the way the world works? Both simulations are simple and both do about as well tracking GNP from 1929 to 1940. We may get some clues to which is better by comparing predictions after 1940. (Data needed to extend the models backward to years prior to 1929 are not readily available.) But it is not clear if the years of World War II are ones in which special relationships hold and thus should be eliminated or if they should be treated as an episode that challenges the predicting ability of the models. Neither simulation, incidentally, predicts with great accuracy after 1940; the Keynesian simulation greatly overestimates GNP during the War because government spending clearly crowds out private spending, and the Quantity Theory simulation misses the leveling off of GNP after the War. In upper-level and graduate courses these issues of validity can be addressed in terms of statistical hypothesis testing and mathematical analysis of models to examine their logic and consistency.[6] Many of the same issues can be treated in a less sophisticated, more intuitive way using these simulations.

Another set of questions raised relates to the period that is simulated: Why did the money stock drop from 1929 to 1933 and in 1937-38? Could changes in business activity influence the level of the money stock? Why did investment move so erratically? Is it reasonable to treat all investment movements as autonomous? Can one explain the international scope of the Depression using only these models? Did political constraints exist which would have prevented effective policy even if it had been tried? Both simulations attempt to raise this last question with comments. In the Keynesian simulation a student usually keeps GNP rising from 1929 to 1933 by increasing the deficit. This causes the computer to respond that newspaper editorials are accusing him of fiscal irresponsibility that will cost him votes on election day. If a student avoids this message by using the balanced budget multiplier, the computer announces a pending tax revolt. In the Quantity Theory simulation a student can force the country off the gold standard by creating too much money.

Finally, if an instructor wishes, he can develop the ISLM model as a compromise between these two extreme models. This is, after all, the way in which the ISLM model was originally presented.

Student Reaction

The two simulations discussed in this paper were presented to students in the Fall, 1978 as part of a package of 55 computerized lessons. This package consisted of a mixture of tutorial lessons, multiple-choice reviews, drills, and simulations.[7] Students of five different principles of macroeconomics sections taught by three different instructors used these lessons. These two simulations were also available in the Fall, 1977 in a less developed form as part of a smaller package of lessons.

After running the simulations, my students wrote short essays explaining what these simulations implied about causes of the Great Depression and what policy recommendations they led to. A major purpose of this assignment was to make students reflect on the models at work in the simulations and then organize their thoughts, steps necessary before they could communicate what they had learned from the simulations. A secondary purpose was to provide feedback regarding effectiveness of the simulations as a teaching device.

The essays revealed that for some students the simulations were very helpful. This group understood the models well and indicated that the exercises helped them understand the differences between textbook monetarist and Keynesian points of view. Other students only partially understood the models underlying the simulations. It was expected that some students would have problems because this exercise asked them to compare competing theories, which involves the highest category in the cognitive domain of Bloom's taxonomy of educational objectives.[8] In addition to students in principles of macroeconomics, some students in upper division courses were asked to complete this exercise. Their essays showed that they had much less difficulty than principles students, and almost all seemed to understand the models involved.

To obtain an objective evaluation of the simulations, students were asked to evaluate them in terms of difficulty, ability to make learning interesting, and overall usefulness. In Fall, 1977, this evaluation was by an in-class questionnaire and in Fall, 1978, it was given by the computer immediately after a student ran a simulation.

TABLE ONE
STUDENT EVALUATION OF THREE SIMULATIONS: AVERAGES, FALL 1978 AND FALL 1977

	E14G	E14H	E15C
Difficulty (1 = very difficult; 5 = very easy)	1.67 1.71	2.38 2.47	2.29 2.35
Ability to Make Learning Interesting (1 = very interesting; 5 = boring)	2.96 3.88	2.29 3.00	2.42 2.88
Overall Usefulness (1 = very useful; 5 = useless)	3.08 3.12	2.33 2.18	2.46 2.06

Number: 1978 = 24; 1977 = 17

Notes: E14G is a simulation written by Keith Lumsden and distributed as GMCR01 by the Hewlett-Packard Contributed Library and as Macroeconomic Policy Game-1 by CONDUIT. E14H is the simulation of the Great Depression using a Keynesian model and E15C is the simulation of the Great Depression using a Quantity Theory model.

The first number of each pair represents the average of responses from Fall, 1978. These data were collected by the computer immediately after a student finished a simulation. Only those students who completed all three simulations are reported, and when a student repeated a simulation, only his last evaluation is used.

The second number of each pair represents the average of responses from Fall, 1977. These data were collected with an in-class questionnaire. Only responses from those who said they completed all three simulations are reported.

Table One displays results from these questionnaires. To provide a comparison, results are also shown for a simulation written by Keith Lumsden of Stanford University which CONDUIT distributes as the simplest of six macroeconomic policy games. (Needless to say, students were not told the source of any of these simulations.) Only results from students who completed all three simulations are shown. These results indicate that the two simulations described in this paper compared favorably to the widely used Lumsden simulation. This conclusion would not be altered by including the many responses from those who completed only one or two of the simulations. It is worth noting that students preferred drills and tutorials to all three simulations. Many students who understood the concepts of fiscal policy and the multiplier when they were used in tutorials or drills found it difficult to apply these concepts in simulations. Often they ran the simulations more than once trying to master them.

Conclusions

This paper pointed out some limitations of using existing macroeconomics simulations for teaching purposes. It suggested that use of simple, real-data simulations had several advantages over existing simulations, one of which was that it opened up questions of validation. Finally, it reported that student reaction to two simple, real-data simulations compared well to reaction to a widely-used, simple, fictional-data simulation.

FOOTNOTES

*This paper was based upon work partially supported by The National Science Foundation under Grant No. SER78-00065. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the author and do not necessarily reflect the views of The National Science Foundation.

1) Thomas R. Kershner, "The Uses of Computer Simulation in Teaching Macroeconomics," Proceedings of a Fourth Conference on Computers in the Undergraduate Curricula, June 18-20, 1973, p. 157.

2) Richard S. Lehman, Computer Simulation and Modeling: An Introduction, (Hillsdale, N.J.: Lawrence Erlbaum Associates, 1977) p. 14.

3) J.M. Dutton and W.H. Starback, "The Plan of the Book," in Dutton and Starback, (Eds) Computer Simulation in Human Behavior (New York: Wiley, 1971), cited in Lehman, p. 14.

4) Milton Friedman and Anna Jacobson Schwartz, A Monetary History of the United States 1867-1960, (Princeton: Princeton University Press, 1963).

5) See Boris P. Pesak, "Monetary Theory in the Post-Robertson 'Alice in Wonderland' Era," Journal of Economic Literature, 14(3), September 1976, pp. 856-9.

6) The Friedman-Meiselman article and the many responses it prompted were arguing a question similar to what these simulations ask — which simple model is a closer approximation to the way the world works. An evaluation of the controversy and references to the principle papers can be found in William Poole and Elinda B.F. Kornblith, "The Friedman-Meiselman CMC Paper; New Evidence on an Old Controversy," American Economic Review, December 1973, pp. 908-17.

7) Slightly more than one third of the lessons were written locally by the author and Dr. Sandra Odorzynski. Major sources for the rest were the University of Notre Dame and the University of Wisconsin-La Crosse. A description of the Notre Dame materials is found in Frank J. Bonello and William Davisson, "Computer Assisted Instruction in Economics at the University of Notre Dame," Proceedings of the 1972 Conference on Computers in Undergraduate Curricula, June 12-14, 1972 (Atlanta, Georgia: Southern Regional Educational Board), pp. 137-54. A listing of the La Crosse material is available in Lawrence A. Daellenbach, et al., CAI Materials for Introductory Macroeconomics

Course, Vol. 2, (La Crosse, Wisconsin: 1977).
8) Benjamin S. Bloom, (Ed.), *Taxonomy of Educational Objectives: Handbook I: Cognitive Domain* (New York: David McKay Comp. Inc., 1956).

SIMULATION #1

```
E14H

THIS SIMULATION USES A SIMPLE KEYNESIAN MODEL, SUCH AS CAN BE FOUND
IN VIRTUALLY ALL PRINCIPLES-OF-MACROECONOMICS TEXTBOOKS, TO EXPLAIN
THE GREAT DEPRESSION.  AS YOU ARE AWARE, IN THE KEYNESIAN MODEL THE
LEVEL OF GNP IS DETERMINED BY THE DEMAND FOR OUTPUT, WHICH IN
TURN IS COMPOSED OF THREE PRIMARY COMPONENTS: CONSUMPTION, INVEST-
MENT, AND GOVERNMENT SPENDING.  THE GOVERNMENT CAN INFLUENCE THE
LEVEL OF GNP WITH FISCAL POLICY, THAT IS, BY CHANGING THE
LEVEL OF GOVERNMENT SPENDING OR BY CHANGING TAXES.

WOULD YOU LIKE TO SEE THE MODEL ON WHICH THIS SIMULATION IS BASED?
YES

GNP = CONSUMPTION + INVESTMENT + GOVERNMENT SPENDING
CONSUMPTION = A + B(GNP - TAXES)
A=14.0,      B=2/3
TAXES ARE SET BY THE POLICYMAKER
INVESTMENT TAKES ITS HISTORICAL VALUES
GOVERNMENT SPENDING IS SET BY THE POLICYMAKER
      (INVESTMENT INCLUDES NET EXPORTS.)


BELOW IS HOW THIS MODEL PREDICTS GNP USING HISTORICAL DATA:

YEAR   CONSUMP-  INVEST-  GOVERNMENT  TAXES  SIMULATED   HISTORICAL
         TION     MENT    SPENDING              GNP         GNP

1929      74.6     17.3         8.8     9.8     100.7       103.4
1930      63.0     10.2         9.5     9.2      82.7        90.7
1931      60.0      6.1         9.5     6.6      75.6        76.1
1932      48.4      1.4         8.3     6.5      58.1        58.3
1933      48.4      1.8         8.2     6.8      58.4        55.8
1934      54.6      3.9        10.0     7.6      68.5        65.3
1935      59.0      6.5        10.2     8.2      75.7        72.5
1936      65.2      8.5        12.2     9.1      85.9        82.7
1937      65.6     12.1        12.0    12.3      89.7        90.7
1938      61.0      7.7        13.2    11.4      81.9        85.0
1939      67.0     10.3        13.5    11.3      90.8        90.8
1940      73.2     14.9        14.2    13.5     102.3       100.0
HISTORICAL GNP IS WHAT ACTUALLY HAPPENED AND SIMULATED GNP IS WHAT
THE SIMPLE KEYNESIAN MODEL USED IN THIS COMPUTER PROGRAM PREDICTS.

YOU WILL NOW HAVE A CHANCE TO SEE IF YOU CAN DO BETTER!  YOUR GOAL
IS TO KEEP GNP GROWING SMOOTHLY AT A MODERATE RATE, ABOUT 3% A YEAR.
TO DO THIS, YOU HAVE THE TWO TOOLS OF FISCAL POLICY AT YOUR DISPOSAL,
THE LEVELS OF GOVERNMENT SPENDING AND TAXES.

ARE YOU READY TO BEGIN
YES

FOR EACH YEAR YOU WILL BE GIVEN THE RESULTS IN THE TABLE ABOVE
AND THEN BE ASKED FOR NEW VALUES OF GOVERNMENT SPENDING AND
TAXES.
```

```
YEAR    CONSUMP-   INVEST-   GOVERNMENT   TAXES   SIMULATED    HISTORICAL
           TION      MENT    SPENDING                GNP           GNP
1929         74.6     17.3           8.8       9.8     100.7 .     103.4

HOW MUCH SHOULD THE GOVERNMENT SPEND  8.8

HOW MUCH SHOULD THE GOVERNMENT TAX  9.8

YEAR    CONSUMP-   INVEST-   GOVERNMENT   TAXES   SIMULATED
            TION      MENT      SPENDING              GNP
1929         74.6     17.3           8.8       9.8     100.7
OVERALL YOU DID A GOOD JOB THIS PAST YEAR

YEAR    CONSUMP-   INVEST-   GOVERNMENT   TAXES   SIMULATED    HISTORICAL
           TION      MENT    SPENDING                GNP           GNP
1930         63.0     10.2           9.5       9.2      82.7        90.7

HOW MUCH SHOULD THE GOVERNMENT SPEND  15.8

HOW MUCH SHOULD THE GOVERNMENT TAX  9.2

YEAR    CONSUMP-   INVEST-   GOVERNMENT   TAXES   SIMULATED
            TION      MENT      SPENDING              GNP
1930         75.6     10.2          15.8       9.2     101.6

YEAR    CONSUMP-   INVEST-   GOVERNMENT   TAXES   SIMULATED    HISTORICAL
           TION      MENT    SPENDING                GNP           GNP
1931         60.0      6.1           9.5       6.6      75.6        76.1

HOW MUCH SHOULD THE GOVERNMENT SPEND  17.5

HOW MUCH SHOULD THE GOVERNMENT TAX  6.6

YEAR    CONSUMP-   INVEST-   GOVERNMENT   TAXES   SIMULATED
            TION      MENT      SPENDING              GNP
1931         76.0      6.1          17.5       6.6      99.6

THE GOVERNMENT DEFICIT THAT YOU PRODUCED WAS 10.9
MANY NEWSPAPER EDITORIALS ARE ACCUSING YOU OF FISCAL
IRRESPONSIBILITY.  THIS MAY COST YOU VOTES ON ELECTION DAY.

YEAR    CONSUMP-   INVEST-   GOVERNMENT   TAXES   SIMULATED    HISTORICAL
           TION      MENT    SPENDING                GNP           GNP
1932         48.4      1.4           8.3       6.5      58.1        58.3

HOW MUCH SHOULD THE GOVERNMENT SPEND  24.3

HOW MUCH SHOULD THE GOVERNMENT TAX  6.5

YEAR    CONSUMP-   INVEST-   GOVERNMENT   TAXES   SIMULATED
            TION      MENT      SPENDING              GNP
1932         80.4      1.4          24.3       6.5     106.1
OVERALL YOU DID A GOOD JOB THIS PAST YEAR

THE GOVERNMENT DEFICIT THAT YOU PRODUCED WAS 17.8
MANY NEWSPAPER EDITORIALS ARE ACCUSING YOU OF FISCAL
IRRESPONSIBILITY.  THIS MAY COST YOU VOTES ON ELECTION DAY.

YEAR    CONSUMP-   INVEST-   GOVERNMENT   TAXES   SIMULATED    HISTORICAL
           TION      MENT    SPENDING                GNP           GNP
1933         48.4      1.8           8.2       6.8      58.4        55.8

HOW MUCH SHOULD THE GOVERNMENT SPEND  58.2

HOW MUCH SHOULD THE GOVERNMENT TAX  56.8
```

```
YEAR   CONSUMP-  INVEST-  GOVERNMENT  TAXES  SIMULATED
           TION     MENT     SPENDING           GNP
1933       48.4     1.8          58.2    56.8    108.4

YOUR CITIZENS ARE NOT HAPPY WITH THE LEVEL OF THE TAXES.

YEAR   CONSUMP-  INVEST-  GOVERNMENT  TAXES  SIMULATED   HISTORICAL
        TION     MENT     SPENDING              GNP         GNP
1934       54.6     3.9          10.0     7.6       68.5        65.3

HOW MUCH SHOULD THE GOVERNMENT SPEND  19.6

HOW MUCH SHOULD THE GOVERNMENT TAX      .6

YEAR   CONSUMP-  INVEST-  GOVERNMENT  TAXES  SIMULATED
           TION     MENT     SPENDING           GNP
1934       87.8     3.9          19.6     0.6    111.3

THE GOVERNMENT DEFICIT THAT YOU PRODUCED WAS 19
MANY NEWSPAPER EDITORIALS ARE ACCUSING YOU OF FISCAL
IRRESPONSIBILITY.  THIS MAY COST YOU VOTES ON ELECTION DAY.

YEAR   CONSUMP-  INVEST-  GOVERNMENT  TAXES  SIMULATED   HISTORICAL
        TION     MENT     SPENDING              GNP         GNP
1935       59.0     6.5          10.2     8.2       75.7        72.5

HOW MUCH SHOULD THE GOVERNMENT SPEND  22.9

HOW MUCH SHOULD THE GOVERNMENT TAX     8.2

YEAR   CONSUMP-  INVEST-  GOVERNMENT  TAXES  SIMULATED
           TION     MENT     SPENDING           GNP
1935       84.4     6.5          22.9     8.2    113.8

THE GOVERNMENT DEFICIT THAT YOU PRODUCED WAS 14.7
MANY NEWSPAPER EDITORIALS ARE ACCUSING YOU OF FISCAL
IRRESPONSIBILITY.  THIS MAY COST YOU VOTES ON ELECTION DAY.

YEAR   CONSUMP-  INVEST-  GOVERNMENT  TAXES  SIMULATED   HISTORICAL
        TION     MENT     SPENDING              GNP         GNP
1936       65.2     8.5          12.2     9.1       85.9        82.7

HOW MUCH SHOULD THE GOVERNMENT SPEND  22.2

HOW MUCH SHOULD THE GOVERNMENT TAX     9.1

YEAR   CONSUMP-  INVEST-  GOVERNMENT  TAXES  SIMULATED
           TION     MENT     SPENDING           GNP
1936       85.2     8.5          22.2     9.1    115.9

THE GOVERNMENT DEFICIT THAT YOU PRODUCED WAS 13.1
MANY NEWSPAPER EDITORIALS ARE ACCUSING YOU OF FISCAL
IRRESPONSIBILITY.  THIS MAY COST YOU VOTES ON ELECTION DAY.

YEAR   CONSUMP-  INVEST-  GOVERNMENT  TAXES  SIMULATED   HISTORICAL
        TION     MENT     SPENDING              GNP         GNP
1937       65.6    12.1          12.0    12.3       89.7        90.7

HOW MUCH SHOULD THE GOVERNMENT SPEND  22

HOW MUCH SHOULD THE GOVERNMENT TAX    12.3
```

```
YEAR   CONSUMP-  INVEST-  GOVERNMENT  TAXES  SIMULATED
          TION     MENT     SPENDING           GNP
1937       85.6    12.1          22.0    12.3     119.7
OVERALL YOU DID A GOOD JOB THIS PAST YEAR


YEAR   CONSUMP-  INVEST-  GOVERNMENT  TAXES  SIMULATED   HISTORICAL
        TION      MENT    SPENDING              GNP         GNP
1938       61.0     7.7          13.2    11.4      81.9       85.0

HOW MUCH SHOULD THE GOVERNMENT SPEND   13

HOW MUCH SHOULD THE GOVERNMENT TAX     13

YEAR   CONSUMP-  INVEST-  GOVERNMENT  TAXES  SIMULATED
          TION     MENT     SPENDING           GNP
1938       57.4     7.7          13.0    13.0      78.1
THE LEVEL OF UNEMPLOYMENT WAS  21.0603       PER CENT
THIS LEVEL OF UNEMPLOYMENT IS UPSETTING THE CITIZENS.
THEY ARE RIOTING IN THE STREETS!!


YEAR   CONSUMP-  INVEST-  GOVERNMENT  TAXES  SIMULATED   HISTORICAL
        TION      MENT    SPENDING              GNP         GNP
1939       67.0    10.3          13.5    11.3      90.8       90.8

HOW MUCH SHOULD THE GOVERNMENT SPEND   32

HOW MUCH SHOULD THE GOVERNMENT TAX     10

YEAR   CONSUMP-  INVEST-  GOVERNMENT  TAXES  SIMULATED
          TION     MENT     SPENDING           GNP
1939      106.6    10.3          32.0    10.0     148.9
THE RATE OF INFLATION YOU CAUSED WAS 33.1951    PERCENT.
A GOOD POLICYMAKER WOULD NOT CAUSE THIS EXCESSIVE INFLATION.
ARE YOU TAKING THE MULTIPLIER PRINCIPLE INTO ACCOUNT?  RECALL
THAT AN ONE DOLLAR INCREASE IN GOVERNMENT SPENDING OR AN ONE
 DOLLAR DECREASE IN TAXES HAS A MULTIPLE EFFECT ON INCOME BECAUSE
IT AFFECTS CONSUMPTION.

THE GOVERNMENT DEFICIT THAT YOU PRODUCED WAS 22
MANY NEWSPAPER EDITORIALS ARE ACCUSING YOU OF FISCAL
IRRESPONSIBILITY.  THIS MAY COST YOU VOTES ON ELECTION DAY.


YEAR   CONSUMP-  INVEST-  GOVERNMENT  TAXES  SIMULATED   HISTORICAL
        TION      MENT    SPENDING              GNP         GNP
1940       73.2    14.9          14.2    13 5     102.3      100.0

HOW MUCH SHOULD THE GOVERNMENT SPEND   15

HOW MUCH SHOULD THE GOVERNMENT TAX     15

YEAR   CONSUMP-  INVEST-  GOVERNMENT  TAXES  SIMULATED
          TION     MENT     SPENDING           GNP
1940       71.8    14.9          15.0    15.0     101.7
THE LEVEL OF UNEMPLOYMENT WAS  21.1452       PER CENT.
THIS LEVEL OF UNEMPLOYMENT IS UPSETTING THE CITIZENS.
THEY ARE RIOTING IN THE STREETS!!

YOUR CITIZENS ARE NOT HAPPY WITH THE LEVEL OF THE TAXES.




YOUR PERFORMANCE IN THIS SIMULATION SHOWED A GOOD UNDERSTANDING
OF THIS MODEL, THOUGH PERHAPS IT COULD BE IMPROVED. KEEP UP THE
GOOD WORK.
```

SIMULATION #2

```
E15C

THE EQUATION OF EXCHANGE AND THE QUANTITY THEORY OF MONEY CAN BE USED
TO EXPLAIN PAST EPISODES OF AMERICAN ECONOMIC HISTORY.  I AM GOING TO
SHOW YOU HOW THE PERFORMANCE OF GNP IN THE YEARS SURROUNDING THE GREAT
DEPRESSION CAN BE SIMULATED USING A SIMPLE QUANTITY THEORY OF MONEY.
AFTER YOU SEE HOW THIS MODEL CAN EXPLAIN THE PERIOD (THOUGH BE AWARE
THAT IT IS NOT THE ONLY EXPLANATION POSSIBLE), I WILL GIVE YOU A CHANCE
TO SHOW HOW WELL YOU UNDERSTAND IT BY MAKING YOU THE MONETARY POLICY
MAKER WITH THE GOAL OF PREVENTING THE DEPRESSION.

WOULD YOU LIKE TO SEE THE MODEL ON WHICH THIS SIMULATION IS BASED
YES

GNP = MONEY STOCK * VELOCITY OF MONEY
MONEY STOCK IS DETERMINED BY THE POLICYMAKER
VELOCITY IS CONSTANT UNLESS THE MONEY STOCK DECLINES,
      IN WHICH CASE IT DROPS AS WELL.


BELOW IS HOW THE MODEL PREDICTS GNP USING HISTORICAL DATA:

YEAR     MONEY   VELOCITY   SIMULATED GNP   HISTORICAL GNP
1929      26.4       3.9         103.0           103.4
1930      25.5       3.7          93.1            90.7
1931      23.5       3.4          79.9            76.1
1932      20.6       3.2          64.9            58.3
1933      19.5       2.9          56.6            55.8
1934      21.5       2.9          62.3            65.3
1935      25.5       2.9          73.9            72.5
1936      29.2       2.9          84.7            82.7
1937      30.3       2.9          87.9            90.7
1938      30.0       2.7          79.5            85.0
1939      33.6       2.7          89.0            90.8
1940      39.0       2.7         103.4           100.0

YOU WILL HAVE A CHANCE TO SEE IF YOU CAN DO BETTER.  YOUR GOAL
IS TO KEEP GNP GROWING SMOOTHLY AT A MODERATE RATE, ABOUT 3% A YEAR.
YOU WILL BE THE SOLE MONETARY POLICYMAKER, ABLE TO SET THE MONEY
STOCK AT WHATEVER LEVEL YOU DECIDE.

ARE YOU READY TO BEGIN  YES

FOR EACH YEAR YOU WILL BE GIVEN THE RESULTS IN THE TABLE ABOVE.
YOU WILL BE ASKED FOR NEW VALUES OF THE MONEY STOCK.

YEAR     MONEY   VELOCITY   SIMULATED GNP   HISTORICAL GNP
1929      26.4       3.9         103.0           103.4

WHAT LEVEL OF MONEY STOCK DO YOU WANT:
27.1
YEAR     MONEY   VELOCITY   SIMULATED GNP
1929      27.1       3.9         105.7
OVERALL YOU DID A GOOD JOB THIS PAST YEAR.

YEAR     MONEY   VELOCITY   SIMULATED GNP   HISTORICAL GNP
1930      25.5       3.7          93.1            90.7

WHAT LEVEL OF MONEY STOCK DO YOU WANT:
27.9
YEAR     MONEY   VELOCITY   SIMULATED GNP
1930      27.9       3.9         108.8
OVERALL YOU DID A GOOD JOB THIS PAST YEAR.

YEAR     MONEY   VELOCITY   SIMULATED GNP   HISTORICAL GNP
1931      23.5       3.4          79.9            76.1

WHAT LEVEL OF MONEY STOCK DO YOU WANT:
28.8
```

```
YEAR    MONEY   VELOCITY   SIMULATED GNP
1931     28.0       3.9          112.3
OVERALL YOU DID A GOOD JOB THIS PAST YEAR.

YEAR    MONEY   VELOCITY   SIMULATED GNP   HISTORICAL GNP
1932     20.6       3.2           64.9             58.3

WHAT LEVEL OF MONEY STOCK DO YOU WANT:
29.5
YEAR    MONEY   VELOCITY   SIMULATED GNP
1932     29.5       3.9          115.1
OVERALL YOU DID A GOOD JOB THIS PAST YEAR

YEAR    MONEY   VELOCITY   SIMULATED GNP   HISTORICAL GNP
1933     19.5       2.9           56.6             55.0

WHAT LEVEL OF MONEY STOCK DO YOU WANT:
30.4
YEAR    MONEY   VELOCITY   SIMULATED GNP
1933     30.4       3.9          118.6
OVERALL YOU DID A GOOD JOB THIS PAST YEAR.

YEAR    MONEY   VELOCITY   SIMULATED GNP   HISTORICAL GNP
1934     21.5       2.9           62.3             65.3

WHAT LEVEL OF MONEY STOCK DO YOU WANT:
31.3
YEAR    MONEY   VELOCITY   SIMULATED GNP
1934     31.3       3.9          122.1
OVERALL YOU DID A GOOD JOB THIS PAST YEAR.

YEAR    MONEY   VELOCITY   SIMULATED GNP   HISTORICAL GNP
1935     25.5       2.9           73.9             72.5

WHAT LEVEL OF MONEY STOCK DO YOU WANT:
32.3
YEAR    MONEY   VELOCITY   SIMULATED GNP
1935     32.3       3.9          126.0
OVERALL YOU DID A GOOD JOB THIS PAST YEAR.

YEAR    MONEY   VELOCITY   SIMULATED GNP   HISTORICAL GNP
1936     29.2       2.9           84.7             82.7

WHAT LEVEL OF MONEY STOCK DO YOU WANT:
28.0
YEAR    MONEY   VELOCITY   SIMULATED GNP
1936     28.0       3.7          102.2
THE LEVEL OF UNEMPLOYMENT WAS  12.5041    PERCENT.
A GOOD POLICYMAKER WOULD NOT CAUSE THIS EXCESSIVE UNEMPLOYMENT

YEAR    MONEY   VELOCITY   SIMULATED GNP   HISTORICAL GNP
1937     30.3       2.9           87.9             90.7

WHAT LEVEL OF MONEY STOCK DO YOU WANT:
27
YEAR    MONEY   VELOCITY   SIMULATED GNP
1937     27.0       3.4           91.8
THE LEVEL OF UNEMPLOYMENT WAS  15.3504    PERCENT.
A GOOD POLICYMAKER WOULD NOT CAUSE THIS EXCESSIVE UNEMPLOYMENT
REMEMBER, YOUR GOAL WAS TO STOP THE GREAT DEPRESSION!!

YEAR    MONEY   VELOCITY   SIMULATED GNP   HISTORICAL GNP
1938     30.0       2.7           79.5             85.0

WHAT LEVEL OF MONEY STOCK DO YOU WANT:
40
```

```
YEAR    MONEY   VELOCITY   SIMULATED GNP
1938       40.0       3.4          136.0
THE RATE OF INFLATION YOU CAUSED WAS  16.734     PERCENT.
WALL-STREET BANKERS BELIEVE THAT THE COUNTRY WOULD BENEFIT BY YOUR
RESIGNATION.
CONGRESS HAS PASSED A RESOLUTION CONDEMNING YOUR POLICIES

YEAR    MONEY   VELOCITY   SIMULATED GNP   HISTORICAL GNP
1939       33.6       2.7          89.0             90.8

WHAT LEVEL OF MONEY STOCK DO YOU WANT!
 55
YEAR    MONEY   VELOCITY   SIMULATED GNP
1939       55.0       3.4          187.0
THE RATE OF INFLATION YOU CAUSED WAS  10.9992    PERCENT.
WALL-STREET BANKERS BELIEVE THAT THE COUNTRY WOULD BENEFIT BY YOUR
RESIGNATION.
CONGRESS HAS PASSED A RESOLUTION CONDEMNING YOUR POLICIES

YEAR    MONEY   VELOCITY   SIMULATED GNP   HISTORICAL GNP
1940       39.0       2.7          103.4            100.0

WHAT LEVEL OF MONEY STOCK DO YOU WANT!
 60
YEAR    MONEY   VELOCITY   SIMULATED GNP
1940       60.0       3.4          204.0
THE RATE OF INFLATION YOU CAUSED WAS  7.34957    PERCENT
```

```
YOUR UNDERSTANDING OF THE MODEL USED IN THIS SIMULATION BORDERS ON
THE INADEQUATE.  YOU NEED TO STUDY IT MORE.
YOU WANTED GNP TO INCREASE BY ABOUT THREE PER CENT A YEAR STARTING FROM
1929, THE LAST YEAR OF PROSPERITY.  SINCE VELOCITY IS A CONSTANT UNLESS
MONEY STOCK FALLS, YOU MUST INCREASE THE MONEY STOCK BY THREE PER
CENT A YEAR EACH YEAR AFTER 1929.  IN 1929 USE THE HISTORICAL MONEY
STOCK.  IN 1930 YOU WANT MONEY TO BE THREE PER CENT HIGHER, OR ABOUT
27.1 (26.4 * 1.03).  IN 1931 YOU WANT MONEY TO BE THREE PER CENT HIGHER
THAN IN 1930, OR 27.1 * 1.03, ABOUT 27.9.  AND SO ON.  TRY IT AND SEE
WHAT HAPPENS.
```

Notes: Both simulations are written in Hewlett-Packard 2000F BASIC. Sources of Data: The National Income and Product Accounts of the United States, 1929-74; Statistical Tables, (Department of Economic Analysis, U.S. Department of Commerce), pp. 324, 339. This publication is a supplement to the Survey of Current Business. Taxes were computed as government expenditures minus surplus or deficit on page 339. Money stock figures were from Lester Chandler, American Monetary Policy, 1929-1941 (Harper Row, 1971). These figures were originally computed by Milton Friedman and Anna Schwartz.

A SMALL INSTRUCTIONAL MACROECONOMIC MODEL OF THE UNITED STATES ECONOMY

Frank M. Werner
Fordham University

For the majority of students, grasping the fundamentals of mathematical model building and use is not an easy task. This is especially so in a graduate school of business where students are admitted to the masters degree program without consideration of their background in abstract mathematics. Too many students cannot translate equations on the blackboard to operational and practical tools, and, at least in the macroeconomic arena, the examples of successful models are so large as to be intractable in a teaching environment.

This paper summarizes the results of a project designed to address the above problem.[1] A small econometric model of the United States economy was specified and estimated. The intent was to make the model as simple as possible to interpret and manipulate, hence the number of equations was deliberately kept to a bare minimum. Variables were limited to (1) those that could be obtained exogenous to the model, (2) those predicted by the model in its simultaneous solution, and (3) a small number that were deemed necessary for the quality of the model. For these latter variables simple one-independent-variable regressions were run to give relationships which were then, reluctantly, added to the model. In each case these added equations have low Durbin-Watson statistics, suggesting the omission of useful explanatory variables; it was decided to accept this limitation for the sake of simplicity. In final form the model consists of thirteen equations.

The Model

It was decided initially to use the Keynesian framework to specify the model. Components of the gross national product were to be modeled; the minimum disaggregation gave six variables:

1. Consumption - non-durables and services (GCNS)[2]
2. Consumption - durables (GCD)
3. Investment - capital goods (GIN)
4. Investment - inventories (GV)
5. Investment - residential housing (GIR)
6. Imports (GIM)

Next variables for interest rates and the price deflator were added:

7. Interest rate - three month Treasury bills (FYGN3)
8. Interest rate - Standard and Poor's AAA bond index (FYAAAC)
9. Price - percent change in GNP implicit price deflator (PDGD)

Then two identities were added to close the model:

10. Gross National Product (Y)

$$Y = C + I + G + X - M$$
$$= (GCNS+GCD) + (GIN+GV+GIR) + G + X - (GIM)$$

where government spending (G) and exports (X) are taken to be exogenous to the model, and

11. Disposable income (GYD)

$$GYD = (1 - \text{tax rate}) \cdot Y$$

where the average tax rate is also an exogenous variable.

As indicated in the introductory note, we had hoped to stop at this point. In two cases, however, the model required an additional independent variable that, it was determined, would be difficult to obtain outside the model; thus two more equations were added:

12. Sales - in the manufacturing sector (MFGSAN)
13. Orders - unfilled backlog (MDU)

The primary source of data was the National Bureau of Economic Research data tapes maintained at the Columbia University Computer Center. Data not on the NBER tapes came from the library of Data Resources, Inc. Data were quarterly, and the model was estimated initially over the period 1960-I to 1972-IV, a total of fifty-two observations. All data were used in real terms, either as available or by dividing by the GNP implicit price deflator for the current quarter.

Five previous econometric models of the United States macro-economy were used as jumping-off points for the specification of the equations. These models were of the Wharton School of the University of Pennsylvania [3], the Brookings Institute [2], Data Resources, Inc. [1], Fair [4] and the Office of Commerce [5]. They were selected to give a cross-section of the more recent and available approaches to model specification within the Keynesian framework. All but the Fair model are quite large; even Fair, who set out to produce a small model, finished with one considerably larger than this one. All have interesting equations for the components of GNP. Each model was studied for the theoretical basis of its equation specification, and a master list of independent variables was compiled. From this were removed duplicates and close substitutes as well as variables which did not fit our objective of limited complexity. Substitute variables were subsequently tested to see if they improved the final equations. The remaining list of explanatory variables formed the basis for the initial regressions.

No record was kept of the number of regressions run; an estimate is in the neighborhood of 500. The RAPE (Regression Analysis Program for Economists) program, available in the systems library of the Columbia University Computer Center, was used; runs were made on the Center's IBM 360/75-91 computer system. All regressions were run using ordinary least squares.

A summary of results follows. t-statistics are enclosed in parentheses directly below each regression coefficient.

Consumption-non-durables and services

```
GCNS = -41.61 + 0.2080 GYD
      (-1.431)  (3.626)

      + 0.6104 GCNSM1
        (6.621)

      + 0.07151 HHSENTM1
        (2.063)

      + 0.0003954 CPOP
        (1.904)
```

Other statistics:

```
 Degrees of freedom                  =47
 Standard error of estimate          = 1.464
 Coefficient of determination(R²)=   .9992
 F(4,47) = 1421.176  →          P =  .000000
 Durbin-Watson statistic             = 2.46
```

The popular formulation of this equation is to make consumption of non-durables and services a function of disposable income (GYD). A better explanatory variable, however, is past consumption as this sets patterns of standard of living; indeed consumption lagged one quarter (GCNSM1) was the most significant variable. Also significant were population (CPOP) and a (University of Michigan) consumer sentiment variable lagged one quarter (HHSENTM1). Increased consumption was hypothesized with population, i.e., more mouths to feed and backs to clothe, and also with sentiment as people tend to spend more when the pressure to save for the future is diminished. All signs were thus expected to be positive.

Consumption - durables

```
GCD = 111.2 - 135.6 GDD + 0.1729 GYD
     (1.802) (-2.714)    (3.951)

      - 4.553 LHURM1 - 0.1999 GCDSTK
        (-8.670)       (-7.013)

      + 0.6793 FMS
        (8.215)
```

Other statistics:

```
 Degrees of freedom                  =46
 Standard error of estimate          = 1.669
 Coefficient of determination(R²)=   .9927
 F(5,46) = 1259.117  →           P=  .000000
 Durbin-Watson statistic             = 1.81
```

The specification of GCD is based upon a stock adjustment model in which the purchase of durables is inversely related to the level of the existing stock. The stock variable (GCDSTK) is constructed from the past fourteen measurements of GCD using a 6% depreciation rate for weights. GDD is a term for the price of durables constructed as the ratio of the price deflator for durables to the GNP implicit price deflator. Inclusion of this term consistently improved the equation and it was thus decided to leave it in even though the price deflator for durables had to be estimated outside the model. High values of GDD should indicate that durables are relatively high priced and result in a lowering of demand. Disposable income (GYD) entered with the expected positive sign as did the M1 money supply (FMS). In addition to directly increasing demand, the money supply variable could also be a proxy for consumer credit since increased money supply should, all things being equal, reduce the interest rate(s). The other significant variable was unemployment lagged one quarter (LHURM1) which, as hypothesized, entered negatively.

Investment - capital goods

```
GIN = -17.26 - 198.6 FYAAAC + 82.83 PX2
     (-2.816) (-4.242)        (6.376)

      + 0.05067 YM1 + 0.2199 IPXMMUM1
        (4.617)       (3.187)
```

Other statistics:

```
 Degrees of freedom                  =47
 Standard error of estimate          = 1.239
 Coefficient of determination(R²)=   .9921
 F(4,47) = 1480.625  →           P=  .000000
 Durbin-Watson statistic             = 1.75
```

Business investment in capital goods is specified as a function of the cost of funds, expectations of expenditures for new plant and equipment, GNP, and the rate of capacity utilization for all manufacturing. The AAA bond rate (FYAAAC) was used to measure the cost of capital, negative sign expected. A positive sign was anticipated for the expectations variable PX2, the expectation of expenditures two quarters ahead. PX1, the coming quarter's expectation was tried and did not perform as well. GNP proved best when lagged one period (YM1) and had the anticipated positive sign. The capacity variable (IPXMMUM1) also entered with the expected positive sign as high capacity utilization should indicate the lack of available facilities for further expansion.

Investment - Inventories

$$GV = \underset{(2.351)}{6.833} + \underset{(4.493)}{0.07985}\,Y + \underset{(6.840)}{0.03497}\,MDU - \underset{(-5.220)}{63.24}\,IVMTM1$$

Other statistics:
Degrees of freedom =48
Standard error of estimate = 2.480
Coefficient of determination(R^2)= .5604
F(3,48) = 20.398 → P= .000000
Durbin-Watson statistic = 1.42

Investment in inventories in this model is a function of GNP (Y), backlog of unfilled orders (MDU), and stock of manufacturing and trade inventories lagged one quarter (IVMTM1). As hypothesized the sign of MDU was positive as high backlog should lead to the production and storage of product to meet this indication of increased demand. IVMTM1, on the other hand, had the anticipated negative sign since high inventory levels reduce the need for additional increases to inventory.

Investment - residential housing

$$GIR = \underset{(1.984)}{2.076} + \underset{(17.78)}{0.8739}\,GIRM1 - \underset{(-4.867)}{0.6593}\,FYGN3 + \underset{(5.490)}{0.00002491}FMDSUM$$

Other statistics:
Degrees of freedom =48
Standard error of estimate = 0.8572
Coefficient of determination(R^2)= .9559
F(3,48) = 259.822 → P= .000000
Durbin-Watson statistic = 1.38

Investment in housing is determined by prior investment in housing and by the cost of money. Since construction usually extends over several quarters, the level of investment of the past quarter (GIRM1) should carry over to form the base for this quarter's investment. The three-month Treasury bill rate (FYGN3) proved the best representative interest rate to measure the cost of capital to the builder and entered with the expected negative sign. Mortgage debt outstanding, as measured by the sum of mortgage debt from savings and loan associations and mutual savings banks (FMDSUM), measured the supply of funds to the buyer and thus entered with a positive sign.

Imports

$$GIM = \underset{(9.987)}{183.3} - \underset{(-11.64)}{18950.}\,PMGD + \underset{(15.65)}{0.1431}\,GYD - \underset{(-5.794)}{34.02}\,MFGSAN$$

Other statistics:
Degrees of freedom =48
Standard error of estimate = 1.874
Coefficient of determination(R^2)= .9769
F(3,48) = 676.579 → P= .000000
Durbin-Watson statistic = 1.46

Imports was specified as a function of the price of imports as a fraction of the implicit price deflator (PMGD), disposable income (GYD), and manufacturing sales (MFGSAN). Imports were anticipated to increase when their relative price declined, hence the negative sign. Higher disposable income should lead to higher consumption of all goods, domestic and imported. The negative sign of the coefficient of manufacturing sales indicates the consumer's trade-off between domestic and foreign goods.

Interest rate - three month Treasury bills

$$FYGN3 = \underset{(5.135)}{3.884} + \underset{(1.695)}{43.84}\,PDGDM1 + \underset{(1.515)}{39.76}\,PDGM2 - \underset{(-5.408)}{0.4441}\,LHURM1 + \underset{(2.765)}{0.002965}\,Y - \underset{(-4.505)}{0.001216}\,FCFRM1$$

Other statistics:
Degrees of freedom =46
Standard error of estimate = 0.4740
Coefficient of determination(R^2)= .8928
F(5,46) = 46.074 → P= .000000
Durbin-Watson statistic = 1.12

The interest rate is determined by the interaction of supply and demand in the money markets; the explanatory variables represent these effects. The level of free reserves lagged one quarter (FCFRM1) represents the supply as the money supply itself in several forms was tried without success. The remaining variables capture the influence of demand. The GNP (Y) and the unemployment rate lagged one quarter (LHURM1) should be direct determinants of loan demand. The rate of growth of the implicit price deflator was introduced to add the impact of inflation to the equation. The one and two quarter lagged variables (PDGDM1 and PDGDM2) worked best in combination even though they both entered with marginal significance.

Interest rate - Standard and Poor's AAA bond index

$$FYAAAC = \underset{(5.928)}{2.404} + \underset{(8.327)}{0.7616}\,FYGN3$$

Other statistics:
Degrees of freedom =50
Standard error of estimate = 0.8983
Coefficient of determination(R^2)= .5810
F(1,50) = 69.343 → P= .000000
Durbin-Watson statistic = 0.14

This relationship was kept as simple and as consistent with the Treasury bill rate as possible.

Price - percent change in GNP implicit price deflator

$$PDGD = \underset{(-1.310)}{-0.002812} - \underset{(-4.633)}{0.1924}\,PGNPD1 - \underset{(-2.428)}{0.02761}\,GNPGAP + \underset{(6.810)}{0.0001674}\,TIME$$

Other statistics:
Degrees of freedom =48
Standard error of estimate = 0.002506
Coefficient of determination(R^2)= .6568
F(3,48) = 30.624 → P= .000000
Durbin-Watson statistic = 1.56

This equation specifies the percent change in the GNP implicit price deflator as a function of the growth rate of GNP (PGNPD1), the gap between potential and actual GNP (GNPGAP), and an index of time (TIME) where 1947-III = 1. A high growth rate of GNP should remove pressure for increased prices as should a high value of the gap, hence the negative signs. The positive sign of the coefficient of TIME suggests the essentially one-way nature of price movements. It is interesting to note that despite some 50 attempts, we could not find a relationship in which unemployment entered with a negative sign, the Phillips curve notwithstanding.

Sales - manufacturing sector

MFGSAN = 0.2107 + 0.001352 Y
(7.154) (25.92)

Other statistics:
Degrees of freedom =50
Standard error of estimate = 0.03651
Coefficient of determination(R^2)= .9307
F(1,50) = 671.593 → P= .00000
Durbin-Watson statistic = 0.16

Orders - unfilled backlog

MDU = 291.8 + 0.4872 - 3.371 GNPGAP
(9.609) (10.79) (-15.02)

Other statistics:
Degrees of freedom =49
Standard error of estimate =30.87
Coefficient of determination(R^2)= .8978
F(2,49) = 215.132 → P= .00000
Durbin-Watson statistic = 0.37

Educational Uses

The model is currently resident on the Fordham University Digital Equipment PDP 11/20 time-sharing system. Available to students are (1) the raw data used to estimate the model[3], (2) an ordinary least squares estimation program[4], and (3) a small program written by the author to accept the exogenous variables as input and to produce a simultaneous solution to the model. Instructional assistance is available to the entire business school faculty with the strongest interest from teachers of Business Economics, Quantitative Methods, and Finance.

Many instructional activities using the model are possible. The choice of assignment depends upon the specific educational objective and the time available. Among those favored by the author and several of his colleagues are the use of the model as a vehicle for:

1. Understanding data sources and data quality --students are asked to locate the source(s) of the various data series and investigate how the data are defined, compiled, summarized, etc. Alternate sources of similar data series are to be compared.
2. Learning the theoretical underpinnings of a model--students are asked to research the theory behind several of the major econometric models and determine how this model agrees or differs.
3. Practicing regression analysis--students are asked to verify the model by re-estimating some or all of the equations and examining the meaning of the accompanying statistics in detail.
4. Considering the effect of updating data--students are asked to extend the data series from 1972-IV to include the most recent observations. They then re-estimate the model and analyze the changes to coefficients and the model's predictive ability.
5. Comparing alternate model specifications--students are asked to research alternate explanatory variables, re-estimate the equations with various combinations of them, and explain the changed performance of the model.
6. Forecasting the macro-economy--students are asked to research values of the exogenous variables and solve the model. Special attention is given to the sensitivity of the solution in response to variations in the input data. Students are asked to identify variables under government control (eg., spending, tax rates) to determine the possible impact of different policy scenarios.

Summary

A small macroeconomic model of the United States economy has proven valuable in making model construction and use a real and hands-on experience for students at the Fordham University Graduate School of Business Administration. The model is easy to understand and use, making it simple for various faculty to incorporate into their courses. The model is small, imperfect, and estimated using data up to but not past 1972, characteristics which encourage students to tinker with it. As such it motivates many students to a level of involvement and understanding that could not be achieved in its absence.

Footnotes

1. The project was initiated under the direction of Professor Richard D. Rippe at the Columbia University Graduate School of Business where the author carried out his doctoral studies. Model specification and estimation were done by the author and Mr. Stephen DiPisa, then a M.B.A. student at Columbia. Implementation and classroom use is ongoing at the Graduate School of Business Administration at Fordham University as part of the Project on Computer Assisted Instruction of which the author is the director.
2. Variable names follow the National Bureau of Economic Research naming system. Lagged variables are identified by the suffix M1 (minus one quarter) for a one period lag and M2 (minus two quarters) for a two period lag.
3. Fordham has the NBER tapes in its data

library so that other data series may be obtained if desired.

4. The more sophisticated student users have access to the entire Fordham systems library which includes Time Series Processor (TSP), Statistical Package for the Social Sciences (SPSS), etc. They can thus apply more advanced estimation procedures, such as two-stage least squares, if desired.

Bibliography

1. Data Resources, Inc., *The Data Resources Econometric Forecasting System, A Preliminary Account,* Unpublished, 1971
2. Duesenberry, Fromm, Klein and Kuh (editors), *The Brookings Quarterly Econometric Model of the United States*, Rand McNally, 1969.
3. Evans, M.K. and L.R. Klein, *The Wharton Econometric Forecasting Model*, Economics Research Unit, Department of Economics, Wharton School of Finance and Commerce, University of Pennsylvania, 1968.
4. Fair, Ray C., *A Short Run Forecasting Model of the United States Economy*, Lexington, Mass., Heath Lexington Books, 1971.
5. Liebenberg, Maurice, Albert Hirsch and Joel Popkin, "A Quarterly Econometric Model of the United States: A Progress Report", *Survey of Current Business*, United States Government Printing Office, May 1966.

AN EXAMINATION OF THE USE OF COMPUTERS TO INCREASE STUDENT MOTIVATION AND UNDERSTANDING IN RELATION TO COMPLEX MATHEMATICAL MODELS

Terry L. Dennis, Rochester Institute of Technology
William J. Stevenson, Rochester Institute of Technology
Francis J. Colella, Simpson College

Courses that deal with the application of quantitative methodology to management decision making, such as management science, operations research, operations management, and quantitative analysis, all have one thing in common--at some point in the course students are required to solve some complex mathematical problems. Hiller and Lieberman[1] relate the successful solution of O.R.-type problems in practice to six basic steps:

1. Formulate the problem
2. Model the system
3. Select a solution technique
4. Obtain a solution to the problem
5. Establish controls over the system
6. Implement the solution

Among the instructors of quantitative courses, there are two groups who deserve attention. One group, surprisingly large, still avoids using the computer as a tool for problem solving, apparently preferring to rely on hand and calculator solutions. The second group use computers in their courses, but only to reduce the burden of number crunching imposed on their students.

With this in mind, the authors initially conducted an experiment designed to measure and evaluate computer versus non-computer assignments dealing with quantitative techniques in terms of: (1) the amount of time spent on homework, (2) the degree of accuracy attained, and (3) the percentage of students completing the assignments. A multiple priority queuing model was used (See Exhibit 1)

EXHIBIT 1.
Multiple Priority Queuing Model

$$W_k = \frac{1}{A\,B_{k-1}\,B_k} + \frac{1}{\mu}$$

where W_k = average waiting time for units in the kth priorty class and

$$B_k = 1 - \frac{\sum_{i=1}^{k} \lambda_i}{M\mu} \quad (B_o = 1)$$

$$A = M! \, \frac{M\,\mu-\lambda}{\rho^M} \sum_{j=o}^{M-1} \frac{\rho^j}{j!} + M\mu$$

$$\text{with } \rho = \frac{\lambda}{\mu} \text{ and } \lambda = \sum_{i=1}^{k} \lambda_i$$

M = number of servers
μ = service rate
k = a priority class

since it is somewhat complex and yet amenable to calculator computations. Students in two sections of an operations management course were given an assignment that required them to determine average waiting time for three priority classes under three differently loaded systems. One section was given access to a computer program that did all computations, the other was not. A quiz was then given to assess differences in performance between the two groups. The results are summarized in Table 1.

Table 1. Performance Characteristics of the Two Groups

	(n=37) Computer	(n=31) Non-Computer
Average time spent on assignment....	24 min.	95 min.
% who completed assignment.......	92%	50%
Accuracy rate of computations.....	100%	33%
Average level of understanding....	63%	5%

Despite the fact that the non-computer students spent four times as long on the assignment as did the computer students, 50% of them failed to complete the assignment. While a few made no attempt, most were simply unable to obtain any sort of solution due to computational difficulties. Even those non-computer students who were able to do the computations achieved only 33% accuracy versus 100% for the computer group; the level of understanding as measured by the quiz was quite low (5%) for the non-computer group, but fairly high for the computer group (63%).

Additional analysis revealed that in non-computer group, only 37.5% of those who completed the first part of the problem attempted additional inputs ($\bar{X}=1.67$), spending almost two hours on the problem. Despite this additional effort, these students still had an extremely low level of understanding. In the computer group, all of the students doing the problem introduced additional inputs ($\bar{X}=7.32$), and achieved a significantly higher level of understanding. Thus, those faculty who contend that students do not develop an in-depth understanding of the solution technique when they use the computer are ignoring the motivational aspects of the situation. Using a modified version of expectancy theory[2] as formulated by Vroom[3] and Porter and Lawler[4], students will be motivated to do homework problems if their expectancy perception (effort-performance probability) and their instrumentality perceptions (performance-reward probabilities) are high. The student must feel that: (1) if he or she attempts to do the problem, there is a high probability that he or she will be able to complete it successfully (high effort-performance probability); and (2) if he or she successfully completes the problem, there will be some positive intrinsic and/or extrinsic reward associated with that completion, e.g. a higher degree of learning, a feeling of accomplishment, a better grade in the course (high performance-reward probability). Hence, when students spend relatively large amounts of time on a problem only to find they either are unable to complete it or complete it incorrectly, they may develop a low expectancy perception and become less motivated to try additional or future problems. Use of a computer model for appropriate problems may, therefore, help to avoid this potential loss of motivation; it may also help to prepare students for the real-world computer codes they are likely to find in practice.

Users whose primary intent is to reduce computational burden fail to recognize some of the more innovative approaches to education. To be sure students of these instructors avoid most of the above-mentioned difficulties. However, it now becomes necessary to deal with the black box syndrome whereby both students and managers have access to vast computational capabilities, particularly in the form of computer packages, which often surpass their understanding. Instead, they need only concern themselves with providing the data input and let the computer hand them the output.

The danger to students is that they will fail to grasp the concepts embodied in the alogrithms; the danger in practice is that decisions are being made by persons who do not really understand the techniques they are using to make those decisions. The problem is real and serious, and must be dealt with. The answer is not to return to non-computerized problem solving, but to use the computers better to train decision makers by focusing attention on the sensitivity of various models to changes in system parameters rather than on simply solving the problem mechanically. Thus, the nature of the assignment ought to be such that instead of merely setting up a problem and obtaining a solution to a given set of conditions, students are required to compare outputs under a variety of system parameters so that they can gain some intuitive appreciation for the degree to which the model will respond to parameter changes. We believe that this will yield an entirely different kind of comprehension of quantitative models, one that will be quite useful in practice.

Accordingly, the authors conducted an investigation of the above premises in a classroom using a series of controlled experiments over a 2-year period with MBA students. The experiments were to measure differences in level of understanding of a complex problem under a variety of formats. In all instances, students obtained solutions using the computer. The formats investigated were:

1. Use the computer to solve a specified problem.
2. Vary system parameters in a predetermined manner and evaluate the sensitivity of the system to such changes.
3. Explore system sensitivity by altering parameters as they see fit.
4. Test hypotheses developed by instructor to judge system sensitivity to parameter changes.
5. Explore parameter changes by developing and testing a series of hypotheses.

RESULTS

Exhibit 2 provides a brief summary of relative understanding attained for the five formats used.

Exhibit 2
Relative Understanding under the Different Formats

Format	Understanding*
1. Solve specific problem	variable (see discussion)
2. Vary parameters in a predetermined way	moderate
3. Student decides how to vary parameters	low to moderate
4. Test hypotheses developed by instructor	moderate to high
5. Students develop and test own hypotheses	low to high

*Rigorous statistical analyses were not attempted due to the relatively large number of independent variables compared to the sample size.

As expected, the accuracy for these computer-based formats was close to 100% in all instances (a few students entered incorrect data). This simply emphasizes the value of computerized computations over manual computations. Effort performance probabilities are always high. It should be noted that on simple calculation type problems (e.g., EOQ), students often shunned the computer model. This was probably because there is a low performance reward probability when the student has a firm grasp of the concept. Of greater interest are the degrees of understanding and related features under the various formats.

When students were given a specified problem to solve, the degree of learning was a function of the amount of time spent, qualtity of class discussion of problems and related background material, as well as the quality of the textbook presentation of the material. Much more time was required for a complex model such as multiple priority queuing than for a less complex model such as exponential smoothing (which was computerized more because of lengthy, repetitious computations than for its complexity).

When students were given a problem and told to vary parameters in a prescribed way, understanding was only moderate, probably because it was possible to complete the entire assignment simply by following instructions; it was not necessary to actually understand why certain parameter values had been selected by the instructor. Understanding was highest when the instructor made an effort to explain the rationale for parameter changes, and lowest when little or no explanation was provided.

When students were allowed to vary parameters as they saw fit, level of understanding varied widely. Some students were completely overwhelmed by the variety of possibilities open to them and so were unable to make satisfactory progress in examining model sensitivity to parameter changes. Others did extremely well.

Students testing hypotheses developed by the instructor fared much better, particularly if the hypotheses were accompanied by a suggested procedure (i.e., parameter changes to try) for evaluating the model. This made the format quite similar to #2, but had the added advantage of introducing the concept of purposeful investigation directed towards testing the hypotheses; the procedure became more definitive. Without suggested changes, the evaluation produced only slightly improved understanding when compared to format #3.

When students were encouraged to develop and then test a series of their own hypotheses, the results again varied widely. Only a few students were unable to make satisfactory progress. Additional time was required of the instructor to teach the students how to formulate and test hypotheses; understanding seemed to be correlated to the amount of instructor effort involved in this. As problem complexity increased and students required additional assistance from the instructor, this format often tended toward format #4. Here the students seemed to feel the least comfortable, and as their expectancy perceptions decreased, the proportion of students undertaking the problem declined somewhat.

CONCLUSIONS

Computer programs provide an excellent means of increasing student understanding of complex mathematical models when they reduce the drudgery and uncertainty of number crunching. In this experiment, they were used only as a supplement to solve problems (although programmed learning concepts could be incorporated into conversational programs). The authors feel that format #5, where the students must develop and test their own hypotheses, offers the students the best chance to learn an approach to problem solving as well as an in-depth understanding of models. Because of the extra time required and the inconsistent results, however, they propose that a combination of formats 4 and 2, where the student is asked to test already developed hypotheses through a suggested procedure, offers the most promising results at this time. It is their intent to test this conclusion further with future classes. It should be stressed that while the use of computer programs in an appropriate manner seems to maintain student motivation, the key to understanding is still the skill of the instructor.

REFERENCES

1 Hillier, Frederick S. and Lieberman, Gerald. Introduction to Operations Research. Holden-Day, 1964.

2 Cummings, L.L., and Schwab, Donald P. Performance in Organizations: Determinants and Appraisal. Scott, Foresman, 1973

3 Vroom, V. H. Work and Motivation. Wiley, 1964.

4 Porter, L.W., and Lawler, E. E. III. Managerial Attitudes and Performance. Dorsey-Irwin, 1968.

THE PROPER ROLE FOR THE COMPUTER SCIENCE PROGRAM AT A SMALL UNDERGRADUATE LIBERAL ARTS UNIVERSITY

Robert L. Horton[1]
Gonzaga University

I Introduction

What is the proper role for the computer science program at a small undergraduate liberal arts university? With such diversity of possibilities and limited resources, how does such a college decide upon the proper scope of its curriculum and activities? Being a four-year school, the program should develop more than just programming skills. These skills are often well taught by the two-year community colleges. But it would be foolish to try to emulate the programs of the resource-rich major graduate institutions. At Gonzaga University in Spokane, our solution was to canvass the local business and industrial firms who would be the potential employers of our graduates and ask them for the skills and knowledge they look for in a new employee. As a result, we found a growing need for a type of training not being provided by educational institutions in our area but which would realize the potential of our four-year program and not overtax our resources. Basically the businesses we contacted are seeking interdisciplinary people, well-versed in both computing and an allied field such as accounting or engineering, and who have sufficient training to communicate with the highly specialized professionals in both areas.

II Description - General

Founded in 1887, Gonzaga University is a Catholic institution sponsored by members of the Society of Jesus (Jesuits) and open to persons of all faiths. The major academic divisions of the university include the College of Arts and Sciences, the School of Business Administration, the School of Education, the School of Engineering, and the School of Law. A lay and Jesuit faculty of about 250 teaches the approximately 2,000 undergraduates and 1,000 law students enrolled. A few graduate degrees are offered but none in the College of Arts and Sciences. The university's operating budget for 1978-79 will be in the vicinity of $10,000,000.

Resources

The Department of Mathematics/Computer Science consists of seven full-time faculty, six of whom have completed their Ph.D.'s within the last seven years. Most of the computer courses are taught under the auspices of this department, primarily by two of the faculty who usually divide their time evenly between computer science and mathematics courses. The standard semester load is four courses, or 12-15 weekly contact hours. There are now 11 computer science courses offered. Usually three to four of these courses per year are handled by adjunct professors to maintain the standard course load for the regular faculty. There are also courses in microcomputers and computer graphics taught by the School of Engineering. See Section IV for a complete program deycription.

The computer center at Gonzaga University presently consists of an IBM 360/30 and peripheral equipment acquired in 1972. This is a single-program, batch only system offering FORTRAN, COBOL, and Assembler languages with 96K of memory. Essentially all academic and administrative work is processed on this machine. The administrative workload is quite heavy including such tasks as registration, financial aid, admissions, student records and affairs, general ledger, payroll, personnel records, and development -- all written in COBOL. The staff at the center consists of a director, assistant director, programmer, operation supervisor, two computer operators, and 1½ fulltime keypunchers, and operates on an annual budget of approximately $155,000.

Philosophy

The university is highly dedicated to the tradition of a liberal arts education. Its mission can perhaps best be described by the following exerpts from the current catalogue describing the College of Arts and Sciences:

"Although the College of Arts and Sciences offers certain programs that include technical training for a particular career, it considers its primary objective to be the formation of its students in the spirit of Christian humanism.... These studies are oriented not only to the pursuit of knowledge but to the growth of the individual through the attainment of that wisdom which recognizes social, moral, and spiritual values.

A core curriculum .... is provided to impart that general culture which distinguishes the educated man or woman and is necessary for higher studies and for the learned professions. This curriculum challenges the student to read, analyze, reflect, and discuss in speech and writing; to weigh the evidence and refrain from hasty judgements; to evaluate with discernment and understanding...."

These are not just flowery phrases designed to impress students and appease alumni, but precisely describe the spirit of the institution and are embodied throughout all of the university's programs including the highly technical specialties.

---

[1] Supported in part by Gonzaga Research Council

How then does a computer science program fit into this liberal arts, humanistic tradition? And is a worthwhile program simply too expensive for a small, private university to afford? Gonzaga is currently considering spending over $250,000, a sizeable portion of its budget, to upgrade its computing hardware. How is it possible to justify an expenditure of this magnitude given the mission of the school and its needs in other areas?

As an institution of higher learning, one of the university's foremost goals is to educate its students to function effectively in modern society. To do so, not only must the student's individual horizons and appreciations be expanded, but he/she must also be given the skills to assure a steady income and to attain a degree of stature within the community. One of the marks of such a person is computer literacy; just as a hundred years ago the mark of a well-educated person was literacy with the written and spoken word. As the computer revolution expands to an ever increasing number of disciplines, job situations, and even our homes, the computer illiterate will find it increasingly difficult to cope personally as well as professionally. Computer work also provides students with an opportunity to exercise and expand their problem formulation, analytic, evaluation and communication skills.

In addition, to remain a viable institution of higher education, a university must continue to provide the physical equipment necessary for a quality liberal education. Today the computer is being compared as an educational resource to the library. A student must master the use of both to acquire a sound education and gain the knowledge and skills necessary to confront the increasingly sophisticated problems of society. With the rapid and complex expansion of knowledge and problem-solving techniques in all areas, computer use is now required in the sciences, business, engineering, mathematics, behavorial sciences, law, and other disciplines. While computers cannot solve all of society's problems alone, most experts recognize that these problems are simply too complex to be solved without the highly sophisticated use of computers.

Within a short time, a university's reputation and eventually its very accreditation will rest in large measure upon the strength and availability of its computing services. A computer-poor university will have trouble recruiting and retaining both faculty and students, while a strong computer program not only will enrich all academic areas but also will provide an additional measure of student, faculty, alumni, and community support.

So to answer the question as to whether or not a college can really afford such a program: the college cannot afford to do without it.

III Business Community Needs

Given that a computer science curriculum is desireable, what is its proper scope? There are and rightfully should be vast differences between the kinds of programs offered by the various types of institutions of higher education. At any particular school, the program should be challenging and yet attainable using available resources.

A number of computer science programs, particularly at small colleges, seem to be floundering because of the lack of a definite role to play or need to fill. Thus the training received by the student is not as effective as it might be.

To determine in a very pragmatic way the proper direction for its program, Gonzaga University interviewed most of the large business concerns, computer vendor district offices, and time-sharing service bureaus in the Spokane area (in addition to reviewing the programs of other universities throughout the region).

One striking fact was that many of the programmers and data processing managers employed by these concerns had little or no formal training in their area beyond that offered by their particular vendor. This was the first indication that higher education was not providing an appropriate training program for at least some segment of the industry.

Lacking contact with academia, many of these people were initially hesitant to meet with the university's representatives and felt threatened discussing the uses and capabilities of their equipment. But after assuring them of our intentions most of them were willing to provide the details of the skills they look for in their personnel.

In this way it was discovered that the business community has a growing need for data processors who can bridge the gap between either the computer vendor's software or the professional programmer and the information needs of business management and engineering. There seems to be a gulf existing between these areas that grows larger as each becomes more specialized. These firms are thus actively seeking personnel who can be the contact between the programmer and the user.

In the Pacific Northwest, there is a great need for computer accountants, especially as more small- to medium-sized firms find the cost of computing within their reach. They discover the accountant and the programmer unable to understand each other's language and problems in sufficient detail to extract from the machine the exact information needed. The accountant describes his problem; the programmer solves the problem as he/she understands it; but the results bear only partial resemblence to what was actually needed. Yet both professionals are quite competent and skilled in their own field. This lack of communication results in delays, internal dissent, a waste of machine time, manpower, and money, and possibly cancellation of worthwhile projects. What is needed is a person with sufficient expertise to communicate and operate professionally in both fields. He then acts as a communications "bus". With faster, more efficient and more error-free information channels, productivity is increased and waste reduced in businesses as well as in computers. In a small firm, an employee may be the programmer; in a larger firm, he/she may play the role of manager, directing the activities of and consulting with the programmers.

Among the necessary skills for this position, businesses listed accounting, finance, statistics, marketing, and management as well as fluency in (usually) three of the following languages -- FORTRAN, COBOL, RPG, BASIC, Assembler (very few other languages were ever mentioned). Many of the firms also asked for a knowledge of file structures and operating systems in addition to the experience of working on a large-scale, complex programming problem without an instructor-fed solution.

Given this need, what type of institution is best equipped to provide this type of training?

The two-year colleges often turn out highly qualified programmers, but they lack the time to develop the necessary depth and breadth. On the other hand, the major graduate institutions are heavily involved in the training of the highly specialized, professional accountants and computer scientists. Their major interest is toward excellence in their own field, leaving little time or interest for interdisciplinary work. This is not a criticism of these schools, but rather an acknowledgement of their right and proper role given their ample resources of staff, equipment, and money.

The small undergraduate university seems to be uniquely adapted to provide the necessary training. It has a full four years to work with the student, but doesn't have to overcome the inertia of deeply entrenched, highly professional programs involving large numbers of students. In this case, the size and inherent friendliness of the school are assets which enhance flexibility. The required interdisciplinary courses can be planned and developed by the various faculty involved who are already familiar with each others program and who have cooperated before in many joint academic endeavors. And the program can easily evolve as needs change.

Thus the small, undergraduate, liberal arts university can have a definite identity in the computer field by creating these interdisciplinary programs which fill a demonstrated and expanding need, can be attained within available resources, and easily fall within the mission of the school.

IV Gonzaga's Program

Until 1976, the computer science course offerings at Gonzaga were limited to a few programming courses. At that time it was decided to expand the curriculum which has now been evolving as new courses have been added, old courses restructured, and the whole scope of the program broadened.

COMPUTER SCIENCE COURSES OFFERED BY THE DEPARTMENT OF MATH AND COMPUTER SCIENCE

CPS 201 -- ELEMENTARY COMPUTER PROGRAMMING -- Elementary FORTRAN.

*CPS 202 -- FORTRAN II -- Advanced structured FORTRAN, inc. programming peripheral devices, internal number rep.

CPS 205 -- COMPUTER SCIENCE FOR BUSINESS -- Elementary COBOL.

*CPS 206 -- DATA PROCESSING -- Advanced structured COBOL, inc. programming peripheral devices, internal number rep.

*CPS 209 -- BASIC -- BASIC programming.

*CPS 302 -- INTRODUCTION TO COMPUTER SCIENCE -- Introduction to the technical aspects of a computer system, inc. compilers, loaders, link editors, intro. to machine language.

CPS 311 -- ASSEMBLER LANGUAGE PROGRAMMING -- Introduction to Assembler.

MAT 350 -- ELEMENTARY NUMERICAL ANALYSIS -- Mathematical applications.

*CPS 403 -- DATA STRUCTURES -- Theory of commonly used data structures including data bases.

*CPS 413 -- SOFTWARE PRINCIPLES -- Study of operating systems.

*CPS 423 -- COMPUTATIONAL TECHNIQUES FOR PROBLEM SOLVING -- Group analysis and solution of large-scale practical computing problems.

* indicates new courses approved during the last three years.

A recent NSF grant has enabled the department to acquire a set of terminals to tie in to Eastern Washington University and offer the 209 course.

In addition, the School of Engineering offers semester courses in analog computing, microprocessors, and computer graphics.

There are currently over 350 students, or approximately 18% of the undergraduates, enrolled annually in computer-based courses, and these figures have been increasing at an annual rate of 12.6% over the past three years -- while total undergraduate enrollment has declined. This increase has occurred despite outmoded equipment. When another regional private university comparable in size recently converted to an on-line, timesharing system, they experienced a fourfold increase in enrollment. Gonzaga expects similar results.

The university does not offer a degree in computer science as such, but does offer a joint Mathematics/Computer Science degree which requires at least four upper division mathematics courses. The degree is relatively new; the initial graduating class of six were granted their degrees just last year (1978). There are now approximately 15 declared majors in the program.

A minor in computer science requires six courses including at least two language courses and the Introduction to Computer Science offering. Students are encouraged to take the upper division courses but may fill the remaining requirements with additional language courses if they are seeking only language skills. This program is only a year old, and since students are not required to declare their minors until petitioning for graduation, there is no way of knowing the numbers involved. But students from other disciplines seeking a knowledge of computing are responsible for the recent enrollment increases.

The computer science curriculum now serves four purposes:

(1) to increase computer literacy

New language courses have been developed and old courses restructured to meet this need. More language courses are planned after the conversion to new equipment.

(2) to provide additional training for community college transfers

A local community college has an excellent two-year programming degree. These students now have an opportunity to enhance and broaden that training.

(3) to prepare majors for graduate study

The upper division courses will adequately prepare the student planning to continue his/her computer studies.

(4) to contribute to interdisciplinary studies

This is the strength and direction of the program. The engineering students naturally gravitate toward computer coursework, finding direct application to their studies, and the best of them schedule as many computer courses as possible.

More and more of the business students are also becoming aware of the importance of computing skills. These students are required to take at least one language course, and these courses have been restructured to include business applications and data processing. In addition, the 423 course allows the better business student to participate in a team solution of a nontrivial business programming problem. The business and computer faculty are also working to include more computer applications in the business curriculum.

The Law School has also become interested in the computer as legal work has become more automated.

V Summary

There is a proper, practical, challenging, and very valuable role that a small, undergraduate, liberal arts university can play in the computer field without overextending itself. By setting up interdisciplinary programs not only among the obvious scientific fields but also in fields, like business, that are becoming increasingly automated, the college is equipping its students not only with the personal skills needed to cope with a highly technical society, but also with job skills that are in great demand by business and industry. This task may be accomplished by merely merging already existing courses of study, but the development of at least a few new courses will usually be necessary. However if successful, substantial enrollment increases can result.

To help in this process, the university should actively seek out contacts with local business and computing firms. They can be a valuable source of practical ideas and directions for guiding a program and providing it with an identity to meet demonstrated needs, as well as a source of moral if not financial support for the approval and implementation of an expanded curriculum. Graduate placement will also be enhanced by these contacts.

A Segmented Structure for Remedial Graduate Computer Science Instruction

Mark Pozefsky
Alan L. Tharp
Computer Science Department
North Carolina State University
Raleigh, North Carolina 27650

The Problem

An element common to most graduate computer science programs is the diversity in backgrounds of students entering such programs. Computers now touch almost every discipline; frequently, a student who is introduced to the computer in an undergraduate curriculum other than computer science becomes enamored with the field and decides to pursue it at the graduate level. Unlike computer science, most disciplines have a relatively limited set of undergraduate majors from which to select their graduate students. For example, chemistry selects the vast majority of its graduate students from chemistry, chemical engineering, physics, and to a lesser extent, other engineering disciplines and mathematics. The wide range of possible undergraduate majors is perhaps unique to graduate computer science education. How can students with such diverse backgrounds who enter graduate computer science programs attain a common level of understanding in their new field of study?

The problem of diverse undergraduate backgrounds has existed since the advent of computer science graduate programs, but it is now compounded by a large number of entering graduate students who do have undergraduate degrees in computer science. Even the students in this group, however, do not have homogeneous backgrounds. Although many undergraduate computer science programs are based upon Curriculum 68 [1], their emphases are different. For example, at North Carolina State University the emphasis is on the student taking a large number of computer science courses to provide a thorough background in computer science; conversely, at the neighboring University of North Carolina at Chapel Hill the orientation is more toward the student taking a small number of computer science courses together with a broad range of courses in related disciplines. Smoliar [2] has observed the effect of these differing approaches on the first-year graduate students at the University of Pennsylvania as have others elsewhere [3,4]. How can students with these differing computer science backgrounds participate in graduate programs together?

A second problem is that many graduate students are now minoring in computer science. They may need remedial work in computer science but have only a limited number of hours to pursue the minor requirement. How can these students efficiently compensate for their deficient backgrounds and still have sufficient time to take the desired graduate level courses?

Finally, a third group of people who desire graduate instruction in computer science, but are not necessarily interested in a degree, are professionals who work with computers but may not have a degree in computer science or if they do, obtained it some time ago. People in this category are often performing a continuing education function. The field of computer science is changing so rapidly that if one does not continue learning, his or her technical knowledge could become obsolete. These professionals may not necessarily want a degree but rather only specialized knowledge in a particular area, e.g. communications theory or data base management systems. How do these individuals easily obtain the necessary background to pursue courses in their particular areas of interest?

Current Solutions

Undergraduate Courses

In departments offering undergraduate programs in computer science, the background deficiencies of individuals in the three groups can be corrected by taking the appropriate undergraduate courses. This procedure does eliminate deficiencies, and until recently was the method used at North Carolina State, but it does have disadvantages.

- The prerequisite chains for a course are often lengthy. Undergraduate courses use a rigid prerequisite structure to avoid chaos and to insure the necessary constituent homogeneity. Both Curriculum 68 [1] and Curriculum 78 [5] exhibit a well-defined prerequisite structure. For a graduate student attempting to eliminate deficiencies, however, the calendar time required to complete the chain of prerequisites may be excessive.
- Undergraduate courses are geared toward individuals being exposed to material for the first time and are presented assuming only an elementary degree of scholastic maturity. The pace of undergraduate courses is therefore often inappropriate for graduate students.

- At many universities, undergraduate courses may not be applied toward a graduate degree.

Special graduate courses

For departments not offering an undergraduate computer science program and also for many who do, a common method to eliminate background deficiencies is a single graduate level course [3,4,6,7], often referred to as an immigration course. These courses cover a multitude of computer science topics ranging from applied algebraic structures to systems programming [e.g. 3]. Quite often such courses are condensations of an entire undergraduate computer science curriculum. Initially, many of these courses were intended to be a single semester, but because of a wealth of material, have expanded into two semester courses (equivalent to six semester hours) [3,4]. While these immigration courses have been very satisfactory in many environments, they have certain weaknesses.

- For students with previous exposure to computer science, a significant portion of these immigration courses will be a duplication. The subject matter dependencies make it difficult for a student to attend only those portions in which he is deficient.
- For a student minoring in computer science at the graduate level, spending six hours on an immigration course may leave insufficient time to pursue a specified interest.
- For working professionals needing course work in only a single area of concentration, a two semester sequence as preparation for only one or two specialized courses may be overkill.

Segmented Structure

A segmented structure consisting of one-hour, limited focus modules is proposed to meet the need presented above and to eliminate many of the weaknesses of current solutions. Each module corresponds roughly to a three-hour undergraduate course considered essential preparation for graduate study in computer science. These modules provide the incoming graduate student with a cafeteria-style offering from which to make the necessary selections. He or she chooses only what he or she needs and no more; he or she may take as many or as few of these modules as he or she wishes. The amount of duplication with previous experience can be minimized.

The present modules (segments) are:

CSE 452: Assembly Language and Basic Computer Organization. Preqs.: high - level programming language.

Basic computer capabilities, basic computer organization, central processing units, main memory, addressing techniques, data representation, error conditions, input, output, symbolic coding, and subroutines.

CSE 453: Data Structures. Preqs : high-level programming language.

Introduction to data structures and the fundamental algorithms associated with their use. Linear lists including stacks, queues, and deques. Sequential and linked allocation of linear lists. Circular lists, arrays amd orthogonal lists. Trees. Garbage collection. Dynamic storage allocation.

CSE 454: Computer Organization and Logic. Preqs.: CSE 452 or equivalent.

Application of Boolean Algebra to combinational circuit design problems. Sequential circuits. Organizational and functional design of simplified computer components such as the memory unit, the arithmetic and logic unit, and input-output devices. Architecture of computing systems. Functional characteristics of I/O devices, data channels, interrupt and priority systems, microprogram control. Hardware-software tradeoffs and firmware.

CSE 455: Applied Algebraic Structures. Preqs.: calculus. high-level programming language.

Mathematical reasoning, sets, binary relations on sets, equivalence relations on sets, equivalence relations and partitions, functions, counting technique, analysis of algorithms, infinite sets, countable and uncountable sets, cardinal arithmetic. Some examples and applications of these algebraic structures are selected from formal language theory, file organization, information retrieval, switching circuits, sequential machines, and artificial intelligence.

CSE 456: Introduction to Computability. Preqs.: CSE 455 or equivalent.

Sequential machines as abstractions of digital computers described by state-transition graphs. Universal Turing machines and halting problem. Church's thesis. Chomsky classification of languages and machines.

These modules are essentially a fast-paced coverage of the corresponding undergraduate courses. Much of the drill, repetitive examples, etc. necessary for undergraduate courses has been eliminated. The modules are to a large extent self-study with formal guidance. The overall prerequisites are only a working knowledge of a high-level programming language and mathematical maturity. Three modules are offered each fall semester (CSE 452, 453 and 455) while the two modules (CSE 454 and 456) that require other modules as prerequisites are offered during the spring semester. Fall scheduling for the majority of the modules allows the entering student to eliminate his or her deficiencies quickly; it causes little hardship on the student since he or she typically requires only one or two of the modules. Each module meets for one 50-minute period per week during a 14 week semester. A module is usually staffed by an instructor who teaches the corresponding undergraduate course so that his additional responsibilities are primarily the extra class meeting per week. The teaching assignments are rotated among a faculty of approximately 20. The grading for the courses is A-B-C-NC. The courses were listed at the 400 level (upper undergraduate-graduate) so that computer science majors would not receive graduate credit for remedial work, but non-major graduate students would (since it would not necessarily be remedial in their cases).

Rationalization

The motivation for the segmented structure is threefold.

- Since students entering a graduate computer science program have diverse backgrounds, every student will not need a survey of all topics to eliminate deficiencies. The segmented approach provides an opportunity for the student to take only those modules that are needed.
- The elapsed time to introduce a topic is spread over a period that is typically three times longer. Instead of lecturing, for example, on machine organization and assembly language 14 times in a five-week period, the segmented structure spreads the same material over a period of 14 weeks. The greater elapsed time gives the student a better opportunity to digest the material. An observation of compressing a 14-week undergraduate course into a five-week summer school course is that even though the students are presented with the same material as in a regular session, they do not grasp the material as well. Certain skills, such as programming, require time to absorb, practice, and perfect.
- A one-hour remedial course, rather than a three-hour one, may encourage someone with only a partially deficient background to take the course rather than picking it up on his or her own. A three-hour course may not seem worth while if the student already knows much of the material. Picking it up on his or her own often becomes not picking it up at all.

Observations

During the fall 1978 semester three immigration modules were taught--assembly language and computer organization (452), data structures (453), and applied algebraic structures (455). Various statistics and observations related to these modules will be given. Although the numbers of students in the courses are too small to allow any statistical inferences, it may be possible to draw conclusions regarding this modular approach to remedial graduate education.

Even though three types of students need something other than the traditional three-hour undergraduate courses, these immigration modules were designed primarily to remediate the computer science (CSE) graduate student. Accordingly, a conscious effort was made to inform all students about the approach to be taken and the workload to be expected. A follow-up warning several weeks into the courses still failed to dislodge several students who should not have been attending. These students were perhaps immune to habitual instructor warnings about course difficulty or maybe were uninterested in the ultimate grade and realized the module was the most time-efficient way to absorb the subject matter. Consequently, the modules have a high level of no credit grades, mostly due to students just leaving the course near the end of the semester.

Every computer science (CSE) graduate student did not get an A even though the modules were directed primarily toward his needs. [With one exception, the grades in 455 were As--the small size (4 students) and homogeneity of purpose (all CSE majors) contributed to a less rigid course and a more lenient grading standard and therefore will not be included in the following statistics.] CSE majors had an overall average of a B- in 452 and 453 (See Figure 1). CSE minors did better in each of the courses with a B+ average. Those considering entering CSE later as either a major or minor did about as well as the CSE minor students (B+ average for both courses). Other students not having any interest in a CSE degree fell between these categories with almost a straight B average. This group had far and away the largest number and percentage of NC grades. Since the primary purpose of these modules is to eliminate deficiencies of CSE graduate students, some unnecessary anxiety may be alleviated (particularly for non-CSE majors) by changing to a pass-fail grading system.

	CSE Major			CSE Minor			Other Students		
	452	453	455	452	453	455	452	453	455
A	1	2	3	2	2		5	5	
B	4	3		1	3		3	4	
C	1	3			1		3	3	
NC		1					1		
NC (faded)	1	1	1				5	3	

Figure 1

Grade Distribution for Modules, Fall 1978.

Since the students with the modular scheme take only the topics they need, there is no clear order of study. Therefore, some material must be repeated throughout all modules. For most of the first-year students, the modules may be their first exposure to the local computing facility, and therefore certain procedural and operational details must be explained during class. Many students don't even know where the computer terminals (interactive or batch reader/printer) are. Once there, they don't know the standard job control formats, data set conventions, or other local pecularities.

Even though the admissions policy requires some proficiency in a high-level programming language, only a minimal amount can be assumed. Valuable lecture or homework time must be devoted to language-specific or programming-related topics. Since no single high-level language is the basis for the curriculum, each course should introduce and use its own meta language, but unfortunately the modules are too short to warrant such additional time.

The overriding factor in all immigration courses is the need to convey a large amount of information in an unreasonably short amount of time. Since we feel our undergraduate courses are, in fact, teaching the proper subject matter, we have tried to pattern the immigration modules after the corresponding undergraduate course. This approach has its advantages and disadvantages.

- Modules are usually taught by someone who has taught the corresponding undergraduate course, so preparation time is decreased. Similar assignments and tests can be used. The instructor has experience with conveying the course concepts, knowing where the trouble spots are, and having a feel for the proper pace.

- Unfortunately, teaching a one-hour module is very much different from doing one-third of a three-hour course. It is not sufficient to leave out two-thirds of each lecture or to talk three times as fast. Less important topics should be eliminated entirely; extra examples and explanatory readings should be left as an exercise for the reader. Trimming the lecture material requires considerable thought about what concepts are most essential and may alter examples and refer-backs during the course.

The modules put considerable responsibility on the students' shoulders. The module's rapid pace gives little time for the student to slough off. There is significant outside reading to reinforce the lectures. Most details, even crucial ones, must be left to examples in the reading or given in handouts.

As mentioned earlier, time is the most important factor in the modules. The instructor must budget the lecture time quite strictly. Unfortunately, this will undoubtedly force him to limit student questions. It is all too easy to let detailed questions overwhelm the lecture period. Student time outside class is also valuable. Because the module only carries one hour credit, the instructor must be careful not to command a disproportionate amount of the student's time. This is not to say that the student might not have to spend the equivalent of a two-credit hour course in terms of homework. However, the instructor must be very selective in terms of written homework and programs, both because of the student's limited time and also the instructor's limited opportunities to evaluate the students. For example, meeting only once a week may preclude a midterm exam. Pop quizzes or small homeworks may not be time-effective either in or out of class. Class time must be devoted to lectures and homework time to broader concepts.

The level of detail necessarily varies quite considerably between modules. The data structures module can be abstract, with only a few concrete programs to demonstrate various implementations. Details are vital and inescapable, when giving the flavor of assembly language. Hand waving will not convey the world of difference between assembler and high-level languages.

Conclusions

The one-credit hour modules to remediate CSE graduate student deficiencies seem to be working well. It will not be possible to fully evaluate the modules' effectiveness, though, until its patrons successfully complete the graduate program, or at least succeeding courses.

Several conclusions can be drawn from the current operation of the modules. Even though billed as for CSE students only, many people from other departments and even from outside the university take the courses. Many drop out, even after completing most of the course, because the work involved is not commensurate with a one-hour of credit. Those who finish appear to have gained a significant amount of knowledge. (In fact, the undergraduate and graduate assembler language classes were given identical final exams. The exam was composed with the undergraduate course (its lectures, assignments, and emphases) in mind. Without analyzing the situation in too much detail, a very crude measure of transmitted knowledge is the average grade. The undergraduates had a 63% average and the graduates a 56% average.)

In some cases, the student's needs would have been better served by taking the standard three-hour undergraduate course, but in others, the difference is warranted. One solution is to introduce a service course for non-CSE students, but our department, like most others, lacks the manpower for extra courses.

The high pressure and self-motivation necessary in the modules is good preparation and a harbinger of the graduate program environment. The student gets early feedback regarding his capacity for graduate level work and can change his or her educational goals before he or she invests considerable time.

The large number of students who enrolled in the first offering of the modules compared to the trickle of non-undergraduate students previously taking the undergraduate courses suggests that the segmented format is preferable to the use of undergraduate courses to eliminate deficiencies. The segmented modules appear to meet a previously unfilled need.

Whether the segmented structure is preferable to a single graduate level course for remediation purposes is more difficult to assess. Based upon our current experience, however, the modular approach is at least a viable alternative, and for the reasons mentioned earlier, it may be preferable in certain situations.

References

1. "Curriculum '68, Recommendations for Academic Programs in Computer Science", CACM, Vol. 11, No. 3, March, 1968, pp. 151-197.
2. Smoliar, Stephen W. "What is a First-Year Computer Science Graduate Student?", ACM/SIGCSE Bulletin, Vol. 8, No. 1, February, 1976, pp. 304-306.
3. Weiner, Leonard H., "Foundations of Computer Science for Entry-level Graduate Students", ACM/SIGCSE Bulletin, Vol. 9, No. 3, August, 1977, pp. 6-9.
4. Mathis, Robert, "A First Course in Computer Science for Graduate Students", ACM/SIGCSE Bulletin, Vol. 10, No. 1, February, 1978, pp. 217-223.
5. Austing et al, "Curriculum '78, Recommendations for the Undergraduate Program in Computer Science", CACM, Vol. 22, No. 3, March, 1979, pp. 147-166.
6. Shaw, Mary, "Immigration Course in Computer Science: 1971 Schedule", ACM/SIGCSE Bulletin, Vol. 4, No. 2, July, 1972, pp. 16-27.
7. Shaw, Mary, "Immigration Course in Computer Science: Teachings Material and 1972 Schedule", ACM/SIGCSE Bulletin, Vol. 5, No. 2, June, 1973, pp. 26-32.

COMPUTER SUPPORT FOR COURSES
in
DATA ENTRY and WORD PROCESSING

by

*Mary Jane Sunkel
Assistant Professor of Business

Jon Todd Rickman.
Director, Computing Services

Jim Hobbs,
Academic Programmer

Northwest Missouri State University
Maryville, Missouri 64468

Telephone: (816)-582-7141

- February 1979 -

*Mrs. Sunkel will make the presentation.

COMPUTER SUPPORT FOR COURSES
in
DATA ENTRY AND WORD PROCESSING

Introduction

One of the fastest growing applications of computer technology is in the creation and revision of written documents (word processing). Providing laboratory facilities for instruction in word processing presents a problem with potentially expensive resource demands; one solution is presented in this paper. One question that must be faced before solving the facility problem is whether or not word processing should be taught, at what level it should be taught, and where it fits into a college or university curriculum. Each institution will have its own financial and historical constraints by which it must be guided, but the authors believe that all schools that have found justification for teaching typing, keypunching, or data entry courses in the past will find even greater justification for offering courses that include word processing.

This paper makes the following distinction between data entry and word processing. Data entry involves record-oriented data which are recorded and then manipulated further by traditional data processing. Word processing records natural language text which results in an end product, the printed document.

Typical curricula in institutions of higher education and advanced vocational education training schools offer one or more degrees which emphasize secretarial skills and which may include data entry using keypunch, key to disc, or key to tape equipment. We believe that such programs soon should be expanded to reflect word processing's expected increase at an annual compound growth rate of better than 30 percent over the next five years.[1]

In the past Northwest Missouri State University has offered courses in typing, office procedures, and keypunching in the School of Business Administration. A keypunch laboratory, including ten IBM 029 keypunches and one IBM 059 verifier, was maintained. Keypunch students also had limited access to two IBM 129 keypunches which were shared with administrative operations. Prior to May 1978, the keypunch equipment was scheduled for six hours a week for the above courses and at other times was available for use by all university students, including undergraduate students in computer science and graduate students. Upon the installation of a PDP 11/70 timesharing computer, need for keypunches by other courses on campus was greatly reduced; therefore, the entire cost of leasing the keypunch equipment became the cost of providing a laboratory for the data entry course. As a result the keypunch class was changed to a data entry class with more varied experiences. After purchasing a software package for word processing and writing several programs to simulate IBM keypunches to run on the timesharing computer, word processing and data entry classes can now be taught with the same terminals used for computer science and other students, again making it possible for all users of computer services to share the same equipment.

---

[1] Viewpoint, International Word Processing Association, Willow Grove, PA, January 1979, p.1.

Description of the Data Entry Course

Northwest Missouri State University has offered a course in card punch training since January 1968. When the course was first offered, only three keypunches, one verifier, and other unit record equipment were available for the instruction of the course. In 1970 ten keypunch machines and one verifier previously scattered around campus were moved to one location, and the general organization of the course became as follows:

- Three sections of the laboratory (limited to a maximum of ten students each) met for two one-hour sessions per week.
- All students in all laboratories met together for two lectures (one hour each) per week.

Lecture (discussion) session topics include:

- general information on the concepts of punched card processing
- details concerning characteristics of cards and data encoding
- organization of data on cards (fields and records)
- differences between IBM card punch machines (026, 029, 129)
- coding data for punching
- planning the most efficient program (drum) cards
- procedures for the proper handling of cards
- good work habits which improve productivity
- various kinds of jobs which use punched cards
- job specifications for laboratory assignments

Laboratories are used strictly as work sessions (all directions are given in the lecture sessions). At the beginning of the semester students work on building skills in numeric punching (knowing the alphabetic keyboard from typewriting was a prerequisite) and the use of the various machine functions (switches and control keys). After these introductory skills are mastered, job assignments begin. A variety of jobs include experience in handling the following:

- numeric, alphabetic, and alphameric fields and considerations for the best way to program each under a variety of conditions
- programmed skip and dup fields
- sequence of fields when different from source document data
- fixed and variable length fields
- use of the left-zero fill-in feature for punching and error correction
- programming in two levels
- coding data while punching
- control punches (such as the X-punch for contra-balances in a numeric field)

Initially all details of the jobs are presented in the lecture session the day before a lab session; students design their program cards before class and then key the program cards and data cards in the lab session.

Later, students do not receive their source document until they arrive at the lab; the assignment is then to study the source document, plan and key the most efficient program card for specified fields, and key the data cards.

Two large jobs (payrolls and invoices) are started in class and completed outside of class. On these jobs students work as partners (one student keying the data, the partner verifying it, and vice versa). The student who does the keying is responsible for making corrections; then the data are reverified. Due to the availability of only one verifier, a sign-up schedule on a bulletin board was necessary for students to plan their time on the verifier.

Toward the end of the semester, sales invoices are used for timing the students in class to arrive at a production grade.

Accuracy is emphasized throughout the semester and a very high grading scale for accuracy is used. The exact grading scale varies somewhat depending upon the job requirements, but a minimum of 90 percent is required for passing.

Data cards on individual assignments are checked by a COBOL program designed and written by the instructor. A set of master cards containing correct data for the job is entered first; the checking program then reads the individual student's cards and compares them with the master deck. If the card is correct, it is tallied and the program continues. If an error is detected, the program output shows a line for the error card followed by a line for the correct master card with a special symbol printed above the columns where mismatches occurred--thus students can easily see their exact errors. A three-digit operator identification number is assigned each student at the beginning of the semester and is punched in all cards that student produces. The identification number is used also in the checking program to determine the number of error cards, the number of correct cards, and the accuracy percentage for each student. Class statistics are summarized by the program, and a grade is listed for each student on each job. At the end of the semester summaries are sorted, and another program averages the grades and determines the semester grade for each student.

In addition to jobs, students are graded on two written tests during the semester, a test on the designing and punching of program cards, and an efficiency rating. Items which affect a student's efficiency rating are:

- promptness (completing assignments and on time);
- good work habits in
  - handling cards,
  - using proper error correction routines,
  - machine control (use of switches and special features);
- design of program cards to make maximum use of machine capabilities;
- following directions;
- overall accuracy (consistency).

Changes in the Data Entry Course

With the installation of the new time-sharing computer, other users on campus no longer needed the use of the keypunch machines; therefore, it was necessary to revamp the course. Students now have the use of only two IBM 029 keypunch machines and limited access to one IBM 129 keypunch. The course now includes more training through the use of CRT terminals and direct interaction with the computer and less training on the keypunch and verifier equipment.

A BASIC program simulating the IBM numeric keyboard arrangement has been written. Students learn this numeric arrangement and develop keying skills by entering data that are compared with a master file containing correct data. When an error is made, a prompt on the screen immediately tells the student to rekey that column. The program accumulates the total number of errors and total keystrokes. When a student reaches a point where a summary is desired, the number of errors, total number of strokes, time used, and percentage of accuracy is indicated on the screen and stored for future reference by the instructor. A similar program allows the student to practice on a standard ten-key numeric arrangement on the auxiliary keypad with the same results being stored.

These two programs have been found to be very effective for building skills and the development of accuracy in numeric keying. The programs appear to have reduced much of the frustration an operator usually has in learning to be accurate on two different ten-key numeric keypad configurations.

Another program used on the terminal allows the student to enter data in answer to prompts on a formatted screen. Invoices, a typical business application, are used for this program. The student enters all data in answer to prompts; data are then stored in a file that can be processed further. The purpose of this program was to introduce the student to interaction computing such as may be found in offices using systems with tailor-made programs for data entry.

Additional Data Entry Software

Simulating a keypunch was considered important even if many students do not use a keypunch after completing the course; if instruction on the keypunch has been mastered, experience can be transferred to other data entry equipment with minor additional training. Many employers still consider keypunching skills an accepted standard on which to continue further training.

KEY is a locally written, interactive BASIC program run from a VT52 DecScope; it simulates an IBM 029 keypunch with some features of a 129 keypunch. A file is created in the user's account with a file name defined by the user. A program (drum) card similar to one for the IBM 029 keypunch is keyed as a single 80-character record. Once entered, a program card is checked to insure that no illegal characters are present; if there are such characters in the card, another program card is requested. When a valid program card is entered, it is stored as the first record of the file. The codes for the program cards are:

0 (zero)	starts auto duplication — first column of field
- (hyphen)	starts auto skip — first column of field
' ' (blank)	numeric field — leave first column blank
A	alpha-numeric field — first column only
&	field continuation character — all columns except first

Keys programmed to simulate 029 special functions are:

[	skip to beginning of next field
^	manual duplication of present field
DELETE	same as ^ key, manual duplication of present field
BACKSPACE	backspace one character at a time
~	backspace to beginning of present field
`	backspace to beginning of present record
^C	exit from entering data to file

KEY allows for the creation of a new file or additions to an existing file. If additions are made to a previously created file, the same program card that was used to create the file is used in extending or adding to the file. The last card (record) entered is displayed on the terminal, and data from it can be used to duplicate any field to the record being entered.

When entering data into a numeric field (using the auxiliary keypad), only zero through nine and spaces are accepted as valid characters; if any other character is entered, a buzzer will sound and the cursor will remain in the same position. Special characters are not accepted as text within the created file and will not print on the screen when typed; only the action for which the key is programmed will take place.

Direct cursor controls are used to leave the terminal screen clear except for the few lines displayed when entering data. A ruler and a counter are always displayed at the top of the terminal screen. The ruler remains constant at all times, with divisions made at zero and every increment of five up to 80; the counter is constantly updated and shows the present position (column) where data will be placed when keyed. Below the ruler, the previous record is displayed directly above the record being entered. These lines are the only lines displayed while entering data records.

When exiting KEY, the name of the newly created file is displayed, and the user is asked if any more files are to be created or extended.

In addition to the direct cursor controls there are four system program calls used within the program. The first allows input of data without carriage returns, the second disables the terminal echo, the third enables terminal echo, and the fourth increases the priority of the running program.

VER is a BASIC program which simulates an IBM 059 verifier and is used after KEY. The same program card that was used and stored in the created file is used again. The same functions and keys are used in VER as were used in KEY.

When executing the VER program, as in KEY, the user is first asked whether or not information on using the program is desired; if so, a short explanation of the keys and their functions are given on the screen, but it is assumed that the student previously has had instruction on the IBM 029 or 059. The next question permits the user to change the file name, thus allowing multiple verifications of the same data. Each verification creates a new file with the corrections made in that file only; the original is never changed. The user is then asked for the name of the file in which the data to be verified is stored and if this file has been partially verified; thus students can verify a large file in small sections as their time permits; no previously verified data are lost.

After these questions are answered, the same type of ruler and counter as in KEY are displayed across the screen. The present record as originally keyed is shown directly below the ruler and the data that is being entered for verification displays on the next line. All checking done in KEY for valid characters in fields is also done in VER. If a mismatch (error) is detected when verifying, the user is given two chances to rekey the same character in the original file; on the third try the new character is accepted and that character replaces the original file character in the new file that is being created as lines are verified. The corrected file has the same name but with a "VER" extension.

VER also employs direct cursor controls, does not require the use of the carriage return, and runs under a raised priority. When all records within a file have been verified, VER will print the name of the completed file and ask if the user wishes to verify another file.

Word Processing Course

Although there is no commonly agreed-upon definition, the authors feel that word processing involves the mechanics of written communications--recording what one has to say on paper, in the desired format, with a high level of accuracy, and with the least possible human effort. Word processing may involve such things as dictating, transcribing, editing, duplicating, filing, mailing, retrieving, and planning the retention or destruction of documents. Many users are emphasizing automated word processing for general correspondence as well as special applications; however, if limitations are necessary, the applications we have found to be the most appropriate for the use of automated word processing equipment are:

1. any time a perfect final copy is desired,
2. any document likely to undergo many revisions before reaching its final form or needing to be periodically updated and reprinted, and
3. any document to be sent to a large number of people and requiring individual addresses or other variable information to be inserted.

A new word processing course, which includes more than just training students how to use equipment, has been taught at NWMSU since September 1978. The first unit of study includes an intensive review of the basic skills (punctuation, capitalization, grammar, spelling, and information) that employers indicate are the weak points of word processing employees; students are then expected to apply these skills properly on all assignments throughout the semester. The following is a general course outline:

2 weeks - review of basic skills (Students are required to purchase a secretarial handbook for use during this review and throughout the semester)

6 weeks - word processing through the computer, including how to create, edit, and print documents, and how to "list process" (merge variable data into letters or other forms)

Specific assignments include abbreviation documents, letters, tabulations, and multiple-page manuscripts.

5 weeks - other equipment:
transcribing machines
duplicating machines
IBM memory typewriter
Olivetti 401 typewriter
IBM model 50 electronic typewriter

Word Processing Software

Computing systems which claim to have word processing software often merely have a text-editor to create a file and another program to format output. These systems are not designed for office personnel but for computer scientists. The software used for word processing should give a dynamic display that always shows the format of the desired printed document.

WORD11[2] (a multi-user, computer-based word processing system) allows for the creation, revision, sorting, and merging of files to create almost any type of document. Input is from a VT-52 or VT-100 DecScope (CRT type terminals).

During the creation and editing of documents, setting margins, spacing, centering, underlining, tabulation, justification, and pagination can be performed as special functions. Corrections and changes can easily be made by deleting one character, word, line, sentence, paragraph, or page at a time. A special key allows for the correction of transpositions without deleting and re-entry. Also, any unit of text can easily be moved from one place to another within a letter or report.

Other functions of WORD11 include: printing documents, deleting documents, changing or modifying names of documents, using abbreviation documents for form paragraphs and frequently used phrases, searching forward and in reverse, and transferring documents from one user's account to another. Spelling error detection is also available. When manuscripts are being entered, footnotes, a table of contents, and an index can be generated by special keys. In short, WORD11 not only performs the functions of a high quality, stand-alone word processor but it also has the advantage of computer data base communications.

Figure 1 shows a menu of WORD11 word processing functions; Figure 2, a menu of print control options; and Figure 3, special key functions.

Hardware and Software Laboratory Costs

Laboratories used for the instruction of data entry and word processing are two of ten terminal laboratories in the academic network on campus. These two laboratories consist of:

1. a single data entry station and a letter quality printer which is in the office machines laboratory (Figure 4);
2. nine scope terminals for data entry, a rough draft printer, a hook up for a portable letter quality printer, and two IBM 29 keypunches (Figure 5).

The total purchase cost of equipment in these two laboratories excluding the two keypunches was:

10 scope terminals @ $1,430	$14,300
1 rough draft printer @$2,100	2,100
1 letter quality printer @$2,900	2,900
Word processing software	7,500
Total	$26,800

Or $2,680 per student data entry station. (This is a one-time cost compared with yearly leasing costs of about $1,000 per keypunch.)

An important fact in evaluating these cost figures is that items one and two are shareable by all users of computing services. Because these items were already necessary for computer science instruction, the cost of the instructional word processing system is $10,400. This cost is still high for total campus usage because at least 20 other terminals are potential users of word processing in addition to the administrative word processing center, which maintains one entry station and two letter quality printers; in fact, it is not unreasonable to split the $10,400 cost over 30 input stations instead of 10. Stand-alone equipment, with which we can make comparisons, has a cost range from $9,000 to $15,000 per station; such a cost would make a structured, monitored laboratory impossible and would limit an instructional laboratory to one or two data entry stations where students would be forced to sign up for individual laboratory experiences.

Conclusion

Data entry and word processing can be taught primarily on interactive computing equipment. A wide range of laboratory experiences are possible on the computer equipment through the use of simulated and formatted data entry programs. Using a timesharing computer makes more student stations possible at a lower cost per station than keypunches, key to tape, key to disc, or stand-alone word processing stations.

Users of administrative applications of word processing have the advantage that they can access admissions, active students, and alumni files for university correspondence.

Plans at Northwest are to expand word processing as a course in the academic office administration program. More concepts will be included in the course so students will be prepared for today's modern office and other situations where they will eventually use many different types of equipment. The choice of a computer-oriented word processing system offers an opportunity for a wider application of word processing principles as well as providing training for more students in a controlled and structured laboratory environment.

---

[2] WORD11 was created and is sold by Data Processing Design, Inc., 6980 Aragon Circle, Suite 8, Buena Park, California 90620.

```
-------------------------------------------------
|                                               |
|                 MAIN MENU                     |
|                                               |
|                                               |
|     C = Create a new document                 |
|     E = Edit an existing document             |
|     P = Print a document                      |
|     I = Index of documents                    |
|     L = List Processing                       |
|     S = Spelling Error Detection              |
|     D = Delete a document                     |
|     U = Printer Utilities                     |
|     T = Document and Transfer Utilities       |
|     F = Finished -- Exit WORD-11              |
|                                               |
|                                               |
|     Type the letter followed by a RETURN      |
|                                               |
-------------------------------------------------
```

Figure 1. WORD11 word processing functions.

```
-----------------------------------------------------------------------------

-- Print Menu --

These are the current settings for printing:

  (1)  Example

NORMAL  Form to print on  (FO)            10  Characters per inch  (PI)
     0  Print priority  (PR)               1  First page to be printed (FR)
   YES  Automatic page breaking  (AP)      0  Last page to be printed  (TO)
    NO  Delete after printing  (DE)        1  Initial page number  (IP)
     1  Copies to print  (CP)             NO  Print header  (PH)
     0  Extra half-line spacing  (EX)     56  Lines per page  (PS)
     0  Print margin indentation  (PM)    NO  Underline spaces  (US)
     6  Lines in top margin  (TM)         NO  Print extra dark  (DA)
     6  Lines in bottom margin  (BM)      NO  Stop every page  (SE)
     0  Wide printing margin  (WM)        NO  Create index  (IN)
    NO  Letter quality printer  (LQ)      NO  Create table of contents  (TC)
                      *None*  Direct Output to  (DT)


     If all the settings are correct, type YES.

     Otherwise, type the letters and value followed by a RETURN.

     Type Gold-MENU to return to the Main Menu without printing

-----------------------------------------------------------------------------
```

Figure 2. Word processing print control options.

```
           SPECIAL GOLD KEY FUNCTIONS FOR WORD PROCESSING

       Main Keyboard                            Auxiliary Keypad

--------------------------------------    ---------------------------
| + Abbreviate          N New Page    |   | Gold | Page | Sent | Ruler|
| B Bottom Document     O Overstrike  |   |--------------------------|
| C Center              P Page Marker|    | < >  | Tab  | UnLn | Del  |
| . Cont Search         ] Print Cntrl|    |--------------------------|
| / Cont Srch/Sel       " Replace     |   | Word | Para | Bold | Cut  |
| D Dead                R Ruler       |   |--------------------------|
| - Figure              , Search      |   | Back | Line | UpCs | Paste|
| F File Document       S Space       |   |--------------------------|
| \ Footnote            A Subscript   |   |             |      |      |
| G Get Document        Q Superscript|    |   Advance   | Sel  | Scrn |
| I Index               T Top Doc     |   |             |      |      |
| K Table of Contents   U UDK         |   |--------------------------|
| L Library             V View        |
| M Menu                X Xtend       |
--------------------------------------
```

Figure 3. WORD11 special key functions.

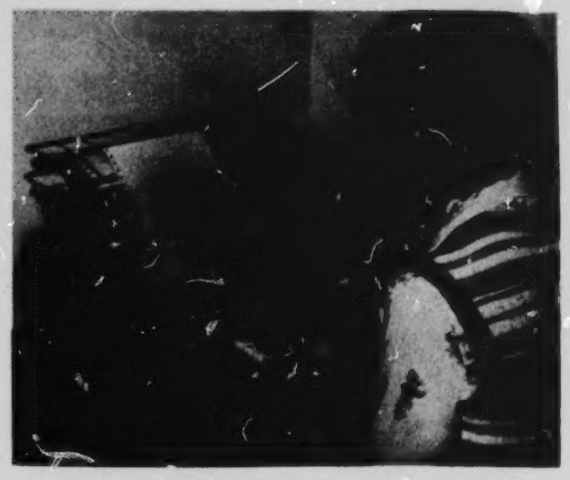


Figure 4. Letter quality output station.

Figure 5. Multiple-function data entry, word processing, and computing services laboratory.

TEACHING DEMOGRAPHY THROUGH THE COMPUTER

Steven Stack
Alma College

Abstract

Demography is one neglected area for applying the computer to the undergraduate sociology curriculum through simulation exercises. The present paper discusses several of the options available under the master demographic program, Demos. Examples are given of both noninteractive and interactive exercises. Many of the simulations are readily adapted to the individual needs of the instructor and students.

The use of the computer in classroom instruction in sociology has gained popularity in recent years (Anderson, 1974, 1975; Conklin, 1976, 1977, 1978; Brown, et al, 1977; Klaff et al, 1977; Erickson and Jacobsen, 1973; Greenblat, 1977; St. George, 1978; Van Valey, 1977; Weiting, 1975). These applications, however, have been disproportionately in the area of data analysis and research methods. Many substantive areas in sociology have been underdeveloped. One neglected area is the simulation of demographic principles and trends. The present paper reviews the utilities of the little-known set of demographic computer simulations, Demos, and gives an example of how these demographic simulations were used to improve instruction in an introductory sociology course.

Demos is a set of noninteractive and interactive demographic computer simulations suitable for time-sharing systems equipped with student-accessible terminals. It is written in FORTRAN, but can be readily adapted to run on systems using BASIC PLUS, a language common to small computers at liberal arts colleges. We adjusted the programs to run in BASIC on our PDP-11/70 computer. Demos is available through contacting the appropriate source at General Electric.[1]

Demos was developed by the General Electric Company under a contract for the Agency for International Development. It took 15 months to develop this fairly sophisticated set of programs. The original objective was to construct some teaching tools to aid planning agencies in developing nations in shaping an understanding of the relationship between demographic and economic variables. Demos is now being incorporated into the curriculum of educational institutions. The set of programs has been adopted by approximately twenty colleges and universities in the United States.

I have been using a number of these Demos exercises, many with my own modifications, for two years to improve instruction in my introductory sociology course in its unit on demography. Many of the subroutines are sufficiently open-ended so that the instructor can write his or her own documentation for creative simulations of demographic processes.[2] The objectives in terms of the demographic principles to be learned can be adjusted according to the student's needs.

Outline of Some Demos Exercises

Exercise 2.1 to 2.3. The first three exercises are noninteractive, they simply print out demographic charts and tables for a time series. They are useful in getting the beginning student accustomed to using the computer terminal, in reducing anxiety, and in conveying a few simple relationships between demographic variables over time. We have found that the beginning student often experiences the most frustration in simply signing onto the computer. It is wise to follow this initial negative experience with a few easy exercises. We have found that if we skip the elementary noninteractive exercises, the students will not perform as well on the more complex, interactive ones.

These exercises all begin with a base population of 10 million. The initial parameters are set to resemble a typical underdeveloped nation. The initial population characteristics of the underdeveloped nation, which we affectionately call UDCLAND, for the year 1975 are:

Gross reproduction rate: 3.049
Life expectancy: 55.0
Infant-child mortality (per thousand births): 139.1
Total population: 10.00 million
Percent Distribution of Population: ages: 0-14: 44.0; 15-49: 45.8; 50+: 10.2
Crude birth rate: 44.1
Crude death rate: 14.0
Population growth rate: 3.0

In order to help the student put these data into perspective we provide corresponding data for the United States. We have also found it helpful either to provide handouts defining the relevant demographic terms or to put the brief definitions up on the computer room blackboard. Otherwise, some students make conceptual errors such as confusing the gross reproduction rate with the rate of population growth.

Exercise 2.1 automatically prints out a 50-year population projection beginning with a high fertility population and under constant fertility and mortality conditions. Population data are printed out in tables and population pyramids, data for each five-year interval on total population, crude birth rate, crude death rate, rate of population growth, and the percent age distribution of the population in three basic age cohorts or age groups (0 - 14, 15 - 49, 50+) are included. More detailed population tables as well as additional population pyramids can be requested which provide data on the number of persons in each of the 16 sex-specific age cohorts. Population grows considerably from 10 million in 1975 to 45.38 million in 2025. Using the printouts as resource material, the students work on problems such as sketching their own projection of what the population pyramid will look like in thirty years.[3]

Exercise 2.2 uses the same initial population and parameters as the previous exercise, except that the infant mortality rate is reduced by 50 percent to 69.6. We point out that this would be analogous to the rapid dissemination of modern pediatric medicine to the underdeveloped nation. The computer print-out provides the same types of data as in the first exercise. Students work out the answers to questions such as: Plot the crude birth rate and gross reproduction rate. Given that only mortality changes, why does the crude birth rate first fall and later rise?

Exercise 2.3 demonstrates the consequences of lowering the gross reproduction rate to half of its previous level, the dream of third world family planner, to 1.527 female children per adult woman. The same categories of information as in previous exercises are automatically printed out by the program. Guided by the appropriate questions, the student can assess the long term consequences of significant reductions in fertility such as its rate of growth and age composition.[4]

Exercise 2.4 represents the first complex, interactive set of programs. There are three options in this exercise. Option one changes the number of persons in any of the 32 sex-specific age cohorts, the survival rates, and/or the fertility rate by age cohorts. Option two enables the student to perform experiments involving percent changes in the initial age composition of the population, its survival rates, and/or its age-specific fertility rates. Option three enables the students to select their own gross reproduction rates, life expectancies, and/or infant mortality rates.

One of the more popular exercises that I have developed from this set of programs is based on option three. I call this the ZPG (zero population growth) exercise. I have the students take on the role of powerful population planners in the capital of UDCLAND. They assume that their subjects have given them unlimited power to curb the impending population problem. The situation is critical. Population must be curbed or there will be mass starvation in the next century. The students manipulate the population parameters so that they can achieve ZPG within 100 years. In essence what they must do is to guide the population through the demographic transition. In addition, they write up a set of practical instructions on how they might accomplish these manipulations--birth control programs, tax breaks for the childless couple, and so on. Most students have to run the program three or four times before they achieve ZPG. They usually make one of two mistakes: they either are too successful and achieve a negative rate of population growth, indicating that the population will become extinct sometime in the future, or they are not daring enough in their manipulations of the parameters and end up with growing populations even after 100-year projections. Although they all fail to complete the exercise the first time around, virtually none of the students complain. Instead they are somewhat delighted to be working with the computer.

Population pyramids are automatically printed for four years representing 25-year intervals. I instruct the students to eyeball the shapes of these pyramids. For example, a pyramid shape means they have been unsuccessful--population is still growing by leaps and bounds. A barrell-shaped pyramid is indicative of ZPG or nearly so. Since it is difficult to achieve an exact ZPG rate, I settle for a rate between -.9 and +.9. Most students can achieve this within 30 minutes of class time. It took me about that much time the first time through as well.

This exercise offers a number of ways of changing the population parameters. The key is to lower the gross reproduction rate to just above one. Many students cut the number of female children per woman to considerably less than one and end up with a negative rate of growth that threatens the survival of the society. Other students achieve ZPG through increasing the infant mortality rate. A few reduce life expectancy. Most attempt some combination of two above. Careful inspection of the population pyramids and other data provide the most reliable solutions. A sample run of this exercise is provided in the appendix.

The students' papers on how they would implement their changes in population parameters in the real world are often quite creative and even bizarre. They have proposed deportation of the elderly, (frightening) infanticide programs to increase the infant mortality rate, tax breaks for those who marry after forty to decrease the gross reproduction rate, and so on.

Many of the exercises are designed so that the instructor can easily write his/her own renditions of population problems for the students. One popular problem I developed from Exercise 2.4 option 1 is what we call the Bubonic Plague Simulation:

> The evil of the plague has fallen on UDCLAND. The survival rate of each cohort will be reduced by a third. However, the International Red Cross has provided all of its miracle vacine which will save 400,000 lives. Simulate this situation using the utilities of Exercise 2.4 option 1. How might the vaccine be distributed to maximize population growth? Offer two plans for distribution and explain why one achieves the goal better than the other.

From this master computer program, which allows for the adjustment of the population in each cohort, survival rates, and fertility rates, I have built exercises where the student uses the canned program to simulate such demographic processes as a baby boom, the long term demographic effects of the immigration of 400,000 children from India, the mass abduction of a large proportion of the country's fertile women, the sudden implementation of health care for the aged and rise in their survival rate, mass starvation, and the impact of war on demographic structures.

The success of computerized instruction can be gauged, in part, by the student evaluation. Over the two years I have been using Demos, in classes from 35 to 70 students, I have found that more than 90 percent of the class say to use it again. The same proportion reported that it helped to develop their understanding of demographic principles. I have found that Demos brings considerable joy to one of the units in introductory sociology which has a potential for being relatively disinteresting to the students.

Notes

[1] For more information on obtaining Demos, the reader is advised to contact:

John Palmisano or George Felman
GE - TEMPO
Room 600
777 14th Street, NW.
Washington, D.C. 20005

[2] A complete list of the exercises available in Demos is given in Appendix 1. Exercises at the 4 to 7 levels are suitable for advanced courses in demography. A demography textbook and guide to the Demos programs, about 100 pages, is available from the Center for Advanced Studies (1974). However, I have found that the documentation for some exercises is repetitive, unimaginative, and/or incomplete. Interested parties might want to read the manual as an introduction to Demos and then go on to constructing their own exercises to go along with the programs.

[3] Other successful questions for the student include: "What does the population profile in 2000 have in common with the profile in 1975?" What changes in age and sex-specific fertility and mortality appear to have occurred over the course of the projection?

[4] Appropriate points of departure for using the exercise to gain an understanding of the relationships between demographic variables include: "Plot the crude birth rate and number of females in each fertile age cohort. Explain why the crude birth rate fails to decrease despite the reduction in the fertility rate. Plot the growth rates for the population in Exercises 2.1 and 2.3. What is the effect of cutting the fertility rate in half on the population growth rate, population size, and the age composition of the population?"

References

Anderson, R.E.
1974 "Bibliography on social science computing," Computing Reviews, July:247-261.

Anderson, R. E.
1975 "Computer based undergraduate curriculum sciences," Teaching Sociology 2:222-223.

Brown, William, Ida Cook, and Charles M. Unkovic
1977 "Effectively teaching undergraduate social research via computer data analysis," In Christensen, Donald (Ed.), Proceedings of the 1977 Conference on Computers in the Undergraduate Curriculum, East Lansing, Michigan p. 323-330.

Center for Advanced Studies
1974 DEMOS: Demographic-Economic Models of Society: A Computerized Learning System, Santa Barbara: General Electric Corporation.

Conklin, George H.
1976 "Using the computer in introductory sociology: a solution to the 'methodology problem'," Teaching Sociology 4(1):83-95.

Conklin, George H.
1977 "Experimental simulation in introductory sociology: modifying exper-sim to meet the needs of beginners," In Christensen, Donald (Ed.), Proceedings of the 1977 Conference on Computers in the Undergraduate Curriculum; East Lansing, Michigan p. 313-322.

Conklin, George H.
1978 "Computer-assisted homework and teaching effectiveness: a case study from introductory sociology," In Prather, Ronald (Ed.), Proceedings of the 1978 Conference on Computers in the Undergraduate Curriculum, Denver, p. 168-179.

Erickson, Maynard and R. Brooke Jacobsen
1973 "On computer applications and statistics in sociology," Teaching Sociology, 1(1):84-102.

Greenboat, Cathy
1977 "Simulation in social science," American Behavorial Scientist, 20(3):411-426.

Klaff, Vivian, Jim Stewart, and John A. Sonquist
1977 "The use of computers in undergraduate sociology," Paper presented at the 72nd annual meeting of the American Sociological Association, Chicago.

St. George, Arthur
1978 "A computer-integrated course in research methods," In Prather, Ronald (Ed.), Proceedings of the 1978 Conference on Computers in the Undergraduate Curriculum, Denver, p. 156-167.

Van Valey, Thomas L.
1977 "The computer and doing sociology," Teaching Sociology, 4(3):277-293.

Weiting, S.
1975 "Simulated research experiences for use in teaching research methods," Teaching Sociology, 3 October: 33-59.

APPENDIX 1: List of Demos Programs

EXCERCISE 2.1
An initial high fertility population is projected for 50 years under constant fertility and mortality conditions.
EXERCISE 2.2
An initial high fertility population is projected for 50 years under constant fertility but reduced infant-child mortality rates.
EXERCISE 2.3
An initial low fertility population is projected for 50 years under constant fertility and mortality conditions.
EXERCISE 2.4
Design your own population projection by using the available options to modify the demographic parameters
EXERCISE 2.5
With selectable specific country data, allows the introduction of time trends in fertility and mortality to explore the concept of momentum
EXERCISE 2.6
Projections to ZPG using selectable specific country data
EXERCISE 3.1 -- TWO PROJECTIONS
1) Constant fertility and gradual decline in mortality
2) Decline in both fertility and mortality
EXERCISE 4.1
Projects student enrollments and teacher requirements (with optional cost data) for a population with constant high fertility.
EXERCISE 4.2
Projects student enrollments and teacher requirements (with optional cost data) for a population with decreased fertility.
EXERCISE 4.3 -- TWO PROJECTIONS
Projects student enrollments and teacher requirements (with optional cost data) for a population with
1) Improved educational and constant demographic parameters
2) Improved educational and changing demographic parameters
EXERCISE 4.4
Allows user modification of both educational and demographic parameters to project student enrollments and teacher requirements.
EXERCISE 4.5 -- TWO PROJECTIONS
Projects student enrollments and teacher requirements (with optional cost cata) for a population with
1) High fertility and infant child mortality
2) High fertility and reduced infant-child mortality
EXERCISE 5.1 -- TWO PROJECTIONS
A 50 year projection of the labor force and work-age populations given
1) Constant high fertility
2) Declining fertility
EXERCISE 5.2
A 50 year projection of the labor force and work-age populations given high fertility and reduced infant-child mortality
EXERCISE 5.3
A projection of the labor force and work-age populations given optional demographic modifications
EXERCISE 6.1 -- TWO PROJECTIONS
50 year economic projections with
1) Moderate marginal propensity to consume
2) Very low marginal propensity to consume
EXERCISE 6.2 -- TWO PROJECTIONS
50 year economic projections with
1) Constant high fertility
2) Declining fertility
EXERCISE 6.3
An economic projection with optional modifications to the economic and demographic parameters
EXERCISE 7.1
Detailed economic projections emphasizing public-private tradeoffs
EXERCISE 7.2
Detailed economic projections emphasizing tradeoffs between education and other government expenditures.
EXERCISE 7.3
Detailed economic projections emphasizing tradeoffs between government investment and other government expenditures.
EXERCISE 7.4
Detailed economic projections emphasizing tradeoffs between education expenditures and government investment.
EXERCISE 7.5
Detailed economic projections emphasizing the interrelated tradeoffs with optional size and composition of the public sector.

APPENDIX 1: List of Demos Programs

```
EXERCISE 2.1
AN INITIAL HIGH FERTILITY POPULATION IS PROJECTED FOR 50 YEARS
UNDER CONSTANT FERTILITY AND MORTALITY CONDITIONS

EXERCISE 2.2
AN INITIAL HIGH FERTILITY POPULATION IS PROJECTED FOR 50 YEARS
UNDER CONSTANT FERTILITY BUT REDUCED INFANT-CHILD MORTALITY RATES

EXERCISE 2.3
AN INITIAL LOW FERTILITY POPULATION IS PROJECTED FOR 30 YEARS
UNDER CONSTANT FERTILITY AND MORTALITY CONDITIONS

EXERCISE 2.4
DESIGN YOUR OWN POPULATION PROJECTION BY USING THE AVAILABLE
OPTIONS TO MODIFY THE DEMOGRAPHIC PARAMETERS

EXERCISE 2.5
WITH SELECTABLE SPECIFIC COUNTRY DATA, ALLOWS THE INTRODUCTION OF TIME
TRENDS IN FERTILITY AND MORTALITY TO EXPLORE THE CONCEPT OF MOMENTUM

EXERCISE 2.6
PROJECTIONS TO ZPG USING SELECTABLE SPECIFIC COUNTRY DATA

EXERCISE 3.1 -- TWO PROJECTIONS
1) CONSTANT FERTILITY AND GRADUAL DECLINE IN MORTALITY
2) DECLINE IN BOTH FERTILITY AND MORTALITY

EXERCISE 4.1
PROJECTS STUDENT ENROLLMENTS AND TEACHER REQUIREMENTS
(WITH OPTIONAL COST DATA) FOR A POPULATION WITH CONSTANT HIGH FERTILITY

EXERCISE 4.2
PROJECTS STUDENT ENROLLMENTS AND TEACHER REQUIREMENTS
(WITH OPTIONAL COST DATA) FOR A POPULATION WITH DECREASED FERTILITY

EXERCISE 4.3 -- TWO PROJECTIONS
PROJECTS STUDENT ENROLLMENTS AND TEACHER REQUIREMENTS
(WITH OPTIONAL COST DATA) FOR A POPULATION WITH
1) IMPROVED EDUCATIONAL AND CONSTANT DEMOGRAPHIC PARAMETERS
2) IMPROVED EDUCATIONAL AND CHANGING DEMOGRAPHIC PARAMETERS

EXERCISE 4.4
ALLOWS USER MODIFICATION OF BOTH EDUCATIONAL AND DEMOGRAPHIC PARAMETERS
TO PROJECT STUDENT ENROLLMENTS AND TEACHER REQUIREMENTS

EXERCISE 4.5 -- TWO PROJECTIONS
PROJECTS STUDENT ENROLLMENTS AND TEACHER REQUIREMENTS
(WITH OPTIONAL COST DATA) FOR A POPULATION WITH
1) HIGH FERTILITY AND INFANT CHILD MORTALITY
2) HIGH FERTILITY AND REDUCED INFANT-CHILD MORTALITY
```

```
EXERCISE 5.1 -- TWO PROJECTIONS
A 50 YEAR PROJECTION OF THE LABOR FORCE AND WORK-AGE POPULATIONS GIVEN
1) CONSTANT HIGH FERTILITY
2) DECLINING FERTILITY

EXERCISE 5.2
A 50 YEAR PROJECTION OF THE LABOR FORCE AND WORK-AGE POPULATIONS GIVEN
HIGH FERTILITY AND REDUCED INFANT-CHILD MORTALITY

EXERCISE 5.3
A  PROJECTION OF THE LABOR FORCE AND WORK-AGE POPULATIONS GIVEN
OPTIONAL DEMOGRAPHIC MODIFICATIONS

EXERCISE 6.1 -- TWO PROJECTIONS
50 YEAR ECONOMIC PROJECTIONS WITH
1) MODERATE MARGINAL PROPENSITY TO CONSUME
2) VERY LOW MARGINAL PROPENSITY TO CONSUME

EXERCISE 6.2 -- TWO PROJECTIONS
50 YEAR ECONOMIC PROJECTIONS WITH
1) CONSTANT HIGH FERTILITY
2) DECLINING FERTILITY

EXERCISE 6.3
AN ECONOMIC PROJECTION WITH OPTIONAL MODIFICATIONS
TO THE ECONOMIC AND DEMOGRAPHIC PARAMETERS

EXERCISE 7.1
DETAILED ECONOMIC PROJECTIONS EMPHASIZING PUBLIC-PRIVATE TRADEOFFS

EXERCISE 7.2
DETAILED ECONOMIC PROJECTIONS EMPHASIZING TRADEOFFS BETWEEN
EDUCATION AND OTHER GOVERNMENT EXPENDITURES

EXERCISE 7.3
DETAILED ECONOMIC PROJECTIONS EMPHASIZING TRADEOFFS BETWEEN
GOVERNMENT INVESTMENT AND OTHER GOVERNMENT EXPENDITURES

EXERCISE 7.4
DETAILED ECONOMIC PROJECTIONS EMPHASIZING TRADEOFFS BETWEEN
EDUCATION EXPENDITURES AND GOVERNMENT INVESTMENT

EXERCISE 7.5
DETAILED ECONOMIC PROJECTIONS EMPHASIZING THE INTERRELATED TRADEOFFS
WITH OPTIONAL SIZE AND COMPOSITION OF THE PUBLIC SECTOR
```

APPENDIX 2: Sample Run of Exercise 2.4, option 3

```
ENTER AN EXERCISE NUMBER
2.4




EXERCISE 2.4
DESIGN YOUR OWN POPULATION PROJECTION BY USING THE AVAILABLE
OPTIONS TO MODIFY THE DEMOGRAPHIC PARAMETERS

DO YOU WANT THE 3 OPTIONS TO EXERCISE C2.4 EXPLAINED? TYPE 1 IF YES, 2 IF NO
?
1

THERE ARE THREE OPTIONS AVAILABLE WITH THIS EXERCISE:
  OPTION 1: TO CHANGE SPECIFIC AGE AND SEX DEMOGRAPHIC
            VARIABLES (BY ABSOLUTE AMOUNTS).
  OPTION 2: TO CHANGE AGGREGATED DEMOGRAPHIC VARIABLES
            (BY PERCENTAGES).
  OPTION 3: TO CHANGE CERTAIN PARAMETERS OF THE RUN
            (AND A FEW DEMOGRAPHIC VARIABLES).


WHICH OPTION IS DESIRED:  1, 2, OR 3 ?
3


HOW MANY YEARS DO YOU WANT THE PROJECTION TO BE? (BETWEEN 10 AND 100
AND DIVISIBLE BY 5)
100

DO YOU WANT TO CHANGE THE DEMOGRAPHIC PARAMETERS? 1=YES, 2=NO
1

INITIAL GRR IS 3.049. WHAT VALUE DO YOU WANT FOR THE FINAL GRR?
1.000
```

```
HOW MANY YEARS TO FINAL GRR? (BETWEEN 0 AND 100)
20

THE INITIAL LIFE EXPECTANCY (FEMALE) IS 55. AND
THE INITIAL INFANT-CHILD MORTALITY RATE IS *****.

  DO YOU WANT TO CHANGE LIFE EXPECTANCY? (TYPE IN 1)
  DO YOU WANT TO CHANGE ONLY THE INFANT-CHILD MORTALITY RATE? (TYPE IN 2)
  DO YOU WANT TO LEAVE MORTALITY UNCHANGED? (TYPE IN 3)
2

WHAT FINAL INFANT-CHILD MORTALITY RATE DO YOU WANT?
(INCLUDE THE DECIMAL POINT)
.050

HOW MANY YEARS TO FINAL INFANT-CHILD MORTALITY RATE? (BETWEEN 0 AND 100)
20


                                    --YEAR--
                         1975        2025       2075
                        ------      ------     ------
                    GRR 3.049       1.000      1.000
        LIFE EXPECTANCY 55.0        60.7       60.7
    INFANT-CHILD MORT. 139.1        50.0       50.0
(PER THOUSAND BIRTHS)



         TOTAL POP.     PERCENT OF POPULATION       AVERAGE FOR 5 YEAR INTERVAL
YEAR     (MILLIONS)     0-14    15-49    50+           CBR      CDR     GROWTH

1975       10.00        44.0    45.8    10.2          44.1     14.0      3.0
1980       11.33        42.5    47.0    10.5          37.2     12.3      2.5
1985       12.53        39.5    49.4    11.1          30.9     10.9      2.0
1990       13.01        32.3    55.3    12.4          16.6      9.0      0.8
1995       13.62        26.8    59.4    13.8          18.3      9.2      0.9
2000       14.28        23.2    61.6    15.3          19.5      9.9      1.0
2005       14.94        24.3    58.7    16.9          19.5     10.5      0.9
2010       15.46        24.4    56.6    19.0          18.0     11.1      0.7
2015       15.81        23.3    55.1    21.6          16.2     11.7      0.4
2020       16.01        21.7    53.5    24.8          14.9     12.5      0.2
2025       16.07        20.3    50.9    28.8          14.3     13.5      0.1
2030       16.03        19.6    48.3    32.1          14.1     14.6     -0.1
2035       15.89        19.5    45.9    34.6          14.1     15.9     -0.2
2040       15.64        19.5    47.0    33.5          14.0     17.2     -0.3
2045       15.27        19.6    47.5    32.9          13.8     18.5     -0.5
2050       14.83        19.6    47.7    32.8          13.6     19.5     -0.6
2055       14.35        19.5    47.5    33.0          13.6     20.1     -0.7
2060       13.90        19.5    47.3    33.3          13.7     20.0     -0.6
2065       13.52        19.5    47.2    33.3          13.8     19.5     -0.6
2070       13.19        19.5    47.1    33.4          13.8     18.7     -0.5
2075       12.85        19.5    47.2    33.3          13.8     19.0     -0.5
```

```
                                  POPULATION PYRAMID
COHORT                1975                                  2005
75+                   MF                                     MFF
70-74                 MF                                     MFF
65-69                 MF                                    MMFF
60-64                MMFF                                  MMMFFF
55-59                MMFF                                 MMMMFFFF
50-54               MMMFFF                               MMMMMFFFFF
45-49              MMMMFFFF                             MMMMMMFFFFFF
40-44              MMMMFFFF                            MMMMMMMFFFFFFF
35-39             MMMMMFFFFF                         MMMMMMMMMFFFFFFFFF
30-34            MMMMMMFFFFFF                       MMMMMMMMMMFFFFFFFFFF
25-29           MMMMMMMFFFFFFF                      MMMMMMMMMMFFFFFFFFFF
20-24         MMMMMMMMMFFFFFFFFF                    MMMMMMMMMMFFFFFFFFFF
15-19        MMMMMMMMMMFFFFFFFFFF                       MMMMMMFFFFFF
10-14      MMMMMMMMMMMMFFFFFFFFFFFF                    MMMMMMMFFFFFFF
 5- 9    MMMMMMMMMMMMMMFFFFFFFFFFFFFF                 MMMMMMMMFFFFFFFF
 0- 4MMMMMMMMMMMMMMMMMMFFFFFFFFFFFFFFFFF             MMMMMMMMMFFFFFFFFF
     $$$$$$$$+$$$$$$$+$$+$$$$$$$+$$$$$$$$$$$$$$$$$$$+$$$$$$$+$$+$$$$$$$+$$$$$
$$$
PERCENT OF   5       1  1       5                   5       1  1       5
TOTAL  POP
```

```
                                  POPULATION PYRAMID
COHORT                2040                                  2075
75+                  MMMFFFF                              MMMMFFFFF
70-74               MMMMFFFFF                             MMMMFFFFF
65-69             MMMMMMFFFFFFF                          MMMMMFFFFFF
60-64            MMMMMMMFFFFFFFF                        MMMMMMFFFFFF
55-59            MMMMMMMFFFFFFFF                        MMMMMMFFFFFF
50-54              MMMMMFFFFF                           MMMMMMFFFFFFF
45-49             MMMMMMFFFFFF                         MMMMMMMFFFFFFF
40-44            MMMMMMMFFFFFFF                        MMMMMMMFFFFFFF
35-39            MMMMMMMFFFFFFF                        MMMMMMMFFFFFFF
30-34            MMMMMMMFFFFFFF                        MMMMMMMFFFFFFF
25-29            MMMMMMMFFFFFFF                        MMMMMMMFFFFFFF
20-24            MMMMMMMFFFFFFF                        MMMMMMMFFFFFFF
15-19             MMMMMMFFFFFF                         MMMMMMMFFFFFFF
10-14             MMMMMMFFFFFF                         MMMMMMMFFFFFF
 5- 9            MMMMMMMFFFFFF                          MMMMMMFFFFFF
 0- 4            MMMMMMMFFFFFFF                        MMMMMMMFFFFFFF
     $$$$$$$$+$$$$$$$+$$+$$$$$$$+$$$$$$$$$$$$$$$$$$$+$$$$$$$+$$+$$$$$$$+$$$$$
$$$
PERCENT OF   5       1  1       5                   5       1  1       5
TOTAL  POP
```

```
DO YOU WISH TO SEE ANOTHER POPULATION PYRAMID? 1=YES, 2=NO
1

SELECT THE YEAR YOU WISH TO SEE (DIVISIBLE BY 5).
1990


                                 POPULATION PYRAMID
COHORT               1990                                  2000

75+                   MF                                    MF

70-74                 MF                                    MF

65-69                MMFF                                  MMFF

60-64                MMFF                                 MMMFFF

55-59               MMMFFF                                MMMFFFF

50-54              MMMMFFFF                              MMMMFFFF

45-49              MMMMFFFF                             MMMMMFFFFF

40-44             MMMMMFFFFF                           MMMMMMFFFFFF

35-39            MMMMMMFFFFFF                        MMMMMMMMFFFFFFFF

30-34           MMMMMMMFFFFFFF                      MMMMMMMMMFFFFFFFFF
25-29         MMMMMMMMMFFFFFFFFF                  MMMMMMMMMMMFFFFFFFFFFF

20-24       MMMMMMMMMMMFFFFFFFFFFF                MMMMMMMMMMMFFFFFFFFFFF

15-19     MMMMMMMMMMMMMFFFFFFFFFFFF               MMMMMMMMMMMFFFFFFFFFFF

10-14     MMMMMMMMMMMMMFFFFFFFFFFFF                   MMMMMMMFFFFFFF

 5- 9      MMMMMMMMMMMMFFFFFFFFFFFF                  MMMMMMMMFFFFFFFF

 0- 4          MMMMMMMMFFFFFFFF                     MMMMMMMMMFFFFFFFFF

***  *******+*******+**+********+******************+*******+**+*******+*****
PERCENT OF  5       1  1        5                  5       1 . 1       5
TOTAL  POP

DO YOU WISH TO SEE ANOTHER POPULATION PYRAMID? 1=YES, 2=NO
2

DO YOU WISH TO SEE DETAILED POPULATION TABLES? 1=YES, 2=NO
1


                        PROJECTED  POPULATION

                              MALE POPULATION
COHORT                        (IN MILLIONS)
          1975   1980   1985   1990   1995   2000   2005   2010   2015   2020
  0- 4    0.88   0.88   0.83   0.50   0.58   0.65   0.68   0.65   0.60   0.56
  5- 9    0.72   0.84   0.83   0.79   0.48   0.55   0.61   0.64   0.62   0.57
 10-14    0.61   0.71   0.83   0.82   0.78   0.47   0.54   0.60   0.63   0.61
 15-19    0.52   0.60   0.70   0.82   0.81   0.77   0.47   0.54   0.60   0.62
 20-24    0.44   0.51   0.59   0.69   0.80   0.80   0.76   0.46   0.53   0.59
 25-29    0.37   0.43   0.50   0.58   0.67   0.78   0.78   0.74   0.45   0.51
 30-34    0.31   0.36   0.42   0.49   0.57   0.66   0.77   0.76   0.72   0.44
 35-39    0.26   0.30   0.35   0.41   0.47   0.55   0.64   0.75   0.74   0.70
 40-44    0.22   0.25   0.29   0.34   0.40   0.46   0.53   0.62   0.72   0.72
 45-49    0.18   0.21   0.24   0.28   0.33   0.38   0.44   0.51   0.60   0.69
 50-54    0.15   0.17   0.20   0.23   0.27   0.31   0.36   0.42   0.48   0.56
 55-59    0.12   0.13   0.16   0.18   0.21   0.24   0.28   0.33   0.38   0.45
 60-64    0.09   0.10   0.12   0.14   0.16   0.19   0.22   0.25   0.29   0.34
 65-69    0.06   0.07   0.09   0.10   0.12   0.13   0.16   0.18   0.21   0.25
 70-74    0.04   0.05   0.06   0.07   0.08   0.09   0.10   0.12   0.14   0.16
 75+      0.04   0.04   0.05   0.05   0.06   0.07   0.08   0.10   0.11   0.13
```

FEMALE POPULATION
(IN MILLIONS)

COHORT	1975	1980	1985	1990	1995	2000	2005	2010	2015	2020
0- 4	0.87	0.86	0.82	0.49	0.57	0.63	0.66	0.64	0.59	0.55
5- 9	0.71	0.83	0.82	0.78	0.47	0.54	0.60	0.63	0.61	0.56
10-14	0.60	0.70	0.82	0.81	0.77	0.47	0.54	0.60	0.63	0.60
15-19	0.51	0.60	0.69	0.81	0.80	0.76	0.46	0.53	0.59	0.62
20-24	0.43	0.51	0.59	0.68	0.80	0.79	0.75	0.45	0.52	0.58
25-29	0.37	0.43	0.50	0.58	0.67	0.78	0.78	0.74	0.45	0.51
30-34	0.31	0.36	0.42	0.49	0.56	0.66	0.77	0.76	0.72	0.44
35-39	0.26	0.30	0.35	0.41	0.47	0.55	0.64	0.75	0.74	0.71
40-44	0.22	0.25	0.29	0.34	0.40	0.46	0.54	0.63	0.73	0.72
45-49	0.18	0.21	0.25	0.29	0.33	0.39	0.45	0.52	0.61	0.71
50-54	0.15	0.17	0.20	0.24	0.27	0.32	0.37	0.43	0.50	0.58
55-59	0.12	0.14	0.17	0.19	0.22	0.26	0.30	0.35	0.41	0.47
60-64	0.10	0.11	0.13	0.15	0.18	0.21	0.24	0.28	0.32	0.38
65-69	0.07	0.09	0.10	0.11	0.13	0.16	0.18	0.21	0.24	0.28
70-74	0.05	0.06	0.07	0.08	0.09	0.11	0.12	0.14	0.17	0.19
75+	0.05	0.05	0.06	0.07	0.08	0.10	0.11	0.13	0.15	0.18

TOTAL POPULATION
(IN MILLIONS)

COHORT	1975	1980	1985	1990	1995	2000	2005	2010	2015	2020
0- 4	1.75	1.74	1.65	1.00	1.15	1.28	1.34	1.29	1.19	1.12
5- 9	1.43	1.66	1.65	1.57	0.95	1.09	1.22	1.27	1.23	1.13
10-14	1.21	1.41	1.65	1.63	1.55	0.94	1.08	1.20	1.26	1.21
15-19	1.03	1.20	1.40	1.63	1.61	1.54	0.93	1.07	1.19	1.24
20-24	0.87	1.02	1.18	1.37	1.60	1.59	1.51	0.91	1.05	1.17
25-29	0.74	0.86	1.00	1.16	1.35	1.57	1.56	1.48	0.89	1.03
30-34	0.62	0.72	0.84	0.97	1.13	1.32	1.53	1.52	1.45	0.87
35-39	0.52	0.60	0.70	0.82	0.95	1.10	1.28	1.49	1.48	1.41
40-44	0.44	0.51	0.59	0.68	0.79	0.92	1.07	1.25	1.45	1.44
45-49	0.36	0.42	0.49	0.57	0.66	0.76	0.89	1.03	1.20	1.40
50-54	0.30	0.34	0.40	0.47	0.54	0.63	0.73	0.85	0.98	1.15
55-59	0.24	0.28	0.32	0.37	0.43	0.50	0.59	0.68	0.79	0.92
60-64	0.19	0.22	0.25	0.29	0.34	0.39	0.46	0.53	0.62	0.72
65-69	0.14	0.16	0.18	0.21	0.25	0.29	0.34	0.39	0.46	0.53
70-74	0.09	0.10	0.12	0.14	0.17	0.19	0.23	0.26	0.30	0.35
75+	0.08	0.09	0.11	0.13	0.15	0.17	0.20	0.23	0.27	0.31

MALE POPULATION
(IN MILLIONS)

COHORT	2025	2030	2035	2040	2045	2050	2055	2060	2065	2070	2075
0- 4	0.54	0.54	0.54	0.53	0.51	0.49	0.47	0.46	0.45	0.44	0.43
5- 9	0.53	0.52	0.51	0.51	0.50	0.48	0.46	0.45	0.44	0.43	0.42
10-14	0.56	0.53	0.51	0.50	0.50	0.49	0.47	0.46	0.44	0.43	0.42
15-19	0.60	0.56	0.52	0.50	0.50	0.50	0.49	0.47	0.45	0.44	0.42
20-24	0.61	0.59	0.55	0.51	0.49	0.49	0.49	0.48	0.46	0.44	0.43
25-29	0.57	0.60	0.58	0.53	0.50	0.48	0.48	0.48	0.47	0.45	0.43
30-34	0.50	0.56	0.59	0.57	0.52	0.49	0.47	0.47	0.47	0.46	0.44
35-39	0.42	0.49	0.55	0.57	0.55	0.51	0.48	0.46	0.46	0.45	0.44
40-44	0.68	0.41	0.47	0.53	0.55	0.53	0.49	0.46	0.45	0.44	0.44
45-49	0.69	0.65	0.39	0.45	0.51	0.53	0.51	0.47	0.44	0.43	0.42
50-54	0.65	0.65	0.62	0.37	0.43	0.48	0.50	0.48	0.45	0.42	0.40
55-59	0.52	0.60	0.60	0.57	0.34	0.40	0.44	0.46	0.45	0.41	0.39
60-64	0.40	0.46	0.54	0.53	0.51	0.31	0.35	0.39	0.41	0.40	0.37
65-69	0.29	0.33	0.39	0.45	0.45	0.42	0.26	0.29	0.33	0.34	0.33
70-74	0.19	0.22	0.25	0.29	0.34	0.34	0.32	0.19	0.22	0.25	0.26
75+	0.15	0.18	0.21	0.24	0.28	0.33	0.34	0.34	0.26	0.24	0.25

FEMALE POPULATION
(IN MILLIONS)

COHORT	2025	2030	2035	2040	2045	2050	2055	2060	2065	2070	2075
0- 4	0.54	0.53	0.53	0.52	0.50	0.48	0.46	0.45	0.44	0.43	0.42
5- 9	0.53	0.51	0.50	0.50	0.49	0.47	0.46	0.44	0.43	0.42	0.41
10-14	0.56	0.52	0.50	0.50	0.50	0.49	0.47	0.45	0.44	0.43	0.42
15-19	0.60	0.55	0.52	0.50	0.49	0.49	0.48	0.46	0.45	0.43	0.42
20-24	0.61	0.59	0.54	0.51	0.49	0.49	0.48	0.47	0.46	0.44	0.43
25-29	0.57	0.60	0.58	0.53	0.50	0.48	0.48	0.47	0.47	0.45	0.43
30-34	0.50	0.56	0.59	0.57	0.52	0.49	0.47	0.47	0.47	0.46	0.44
35-39	0.43	0.49	0.55	0.57	0.55	0.51	0.48	0.46	0.46	0.45	0.45
40-44	0.69	0.42	0.48	0.53	0.56	0.54	0.50	0.47	0.45	0.45	0.44
45-49	0.70	0.67	0.40	0.46	0.52	0.54	0.52	0.48	0.45	0.44	0.43
50-54	0.68	0.68	0.64	0.39	0.45	0.50	0.52	0.50	0.46	0.43	0.42
55-59	0.55	0.64	0.64	0.61	0.37	0.42	0.47	0.49	0.48	0.44	0.41
60-64	0.44	0.51	0.59	0.59	0.56	0.34	0.39	0.43	0.45	0.44	0.40
65-69	0.33	0.38	0.45	0.52	0.52	0.49	0.30	0.34	0.38	0.40	0.38
70-74	0.22	0.26	0.30	0.35	0.41	0.41	0.39	0.23	0.27	0.30	0.31
75+	0.21	0.24	0.28	0.32	0.38	0.44	0.46	0.46	0.35	0.33	0.34

```
                              TOTAL POPULATION
COHORT                         (IN MILLIONS)
        2025  2030  2035  2040  2045  2050  2055  2060  2065  2070  2075

  0- 4  1.08  1.07  1.06  1.04  1.01  0.97  0.93  0.91  0.89  0.87  0.84
  5- 9  1.06  1.03  1.02  1.01  0.99  0.96  0.92  0.89  0.87  0.85  0.83
 10-14  1.12  1.05  1.01  1.00  1.00  0.98  0.94  0.91  0.88  0.86  0.84
 15-19  1.20  1.11  1.04  1.00  0.99  0.99  0.97  0.93  0.90  0.87  0.85
 20-24  1.22  1.18  1.09  1.02  0.99  0.98  0.97  0.95  0.92  0.88  0.85
 25-29  1.15  1.20  1.16  1.07  1.00  0.97  0.96  0.95  0.93  0.90  0.86
 30-34  1.01  1.12  1.17  1.13  1.04  0.98  0.95  0.93  0.93  0.91  0.88
 35-39  0.85  0.98  1.09  1.14  1.10  1.02  0.95  0.92  0.91  0.91  0.89
 40-44  1.37  0.83  0.95  1.06  1.11  1.07  0.99  0.93  0.90  0.89  0.88
 45-49  1.39  1.32  0.80  0.92  1.02  1.07  1.03  0.95  0.89  0.86  0.85
 50-54  1.33  1.33  1.26  0.76  0.88  0.98  1.02  0.98  0.91  0.85  0.82
 55-59  1.07  1.25  1.24  1.18  0.71  0.82  0.91  0.95  0.92  0.85  0.80
 60-64  0.83  0.97  1.13  1.12  1.07  0.64  0.74  0.82  0.86  0.83  0.77
 65-69  0.62  0.71  0.83  0.97  0.96  0.91  0.55  0.64  0.71  0.74  0.71
 70-74  0.41  0.46  0.55  0.64  0.75  0.75  0.71  0.43  0.49  0.55  0.57
 75+    0.36  0.42  0.49  0.56  0.66  0.76  0.80  0.80  0.61  0.58  0.60

DO YOU WISH TO RUN ANOTHER EXERCISE? 1=YES, 2=NO
1
```

THE USE OF CENSUS DATA ANALYSIS TO ENHANCE THE SOCIOLOGY CURRICULUM

by

Mr. Harold Benenson
Dept. of Social Relations
Ramapo College of N.J.
505 Ramapo Valley Road
Mahwah, New Jersey 07430

Dr. Steven Just
Academic Services Dept.
N.J. Educational Computer Network
P.O. Box 390
New Brunswick, N.J. 08903

Introduction

It has been realized for many years that the properly trained social scientist should have a working knowledge of the use of computers for data analysis. At most colleges and universities, one or more courses in research methods have been incorporated into the social science curriculum. Usually, the fundamental ingredient of these courses is the analysis of several data sets via a standard statistical package (such as SPSS). In some courses, students design survey instruments, gather the data, establish hypotheses, and test these hypotheses using the computer to perform the appropriate statistics. In other courses, the data set to be analyzed is pre-packaged and, the student merely performs the analysis. Either method is appropriate, depending upon the course objectives. While in the latter courses the instructor may wish to concentrate on statistics, computing, and data analysis, the objective of the first type of course may be to introduce the student to the full range of analysis techniques, from instrument design to final conclusions.

Whatever the method chosen, one of the goals of any course of this type is to use real data. It is appropriately felt that to ensure maximum benefit and to keep the student's interest, it is desirable to introduce data that the student may encounter in the real world. Both the student-gathered data method and the pre-packaged data method have advantages and disadvantages when judged on this criterion. When the student gathers his own data, certainly the data are real and probably relevant to the student. Due to time constraints, however, the sample size is usually quite small and often the scope of the data is trivial. Using pre-packaged data allows the student to analyze large data sets with many variables per case. These data sets, however, often have the disadvantage of being somewhat remote and irrelevant to the student.

In response to these problems, sociology faculty at Ramapo College of New Jersey have initiated a project to use data from the 1970 Census in the classroom. Certainly the census data meet the criterion of scope, depth, and relevance. The problems encountered with the use of Census data are cost and the feasibility of introducing such a large and complex data set into the undergraduate curriculum. The cost issue has been solved by purchasing only a subset of the census data. Through careful planning the analysis of census data has been introduced into courses at both the freshman and senior levels.

Census Data

Ramapo College is one of the eight State Colleges of New Jersey and one of the principal members of the New Jersey Educational Computer Network (NJECN), a private, not-for-profit consortium established by the State of New Jersey to provide computing services to higher education. It has long been the goal of NJECN to obtain census data for instructional and research use by faculty at its member institutions. This goal has been frustrated by the expense of acquisition, which can run to several thousand dollars. During the 1977-78 academic year, it was suggested that only a subset of the entire census need be purchased.

Basically, the census consists of information derived from questions asked of the entire population or a sample of the population. One type of sample questionnaire went to 15% of all households; the other went to 5% of all households. The data from these questionnaires are distributed in several forms. They are released as summary tables in printed reports, as summary data on computer tape, and as public use samples.

The printed reports can be found in most academic libraries and obviate the need for computers. They restrict the researcher, however, to the investigation of only those questions for which summary tables have been produced. The summary data on tape have the same kind of data found in the printed reports--just more of it. There are six major series of summary tapes, called counts, involving about 2,000 reels of tape if all files were obtained.

We did not purchase any summary tapes because they would not allow the students to program their own analyses. Instead, we purchased tapes that contained the raw census data; these are known as Public Use Samples (PUS). Six Public Use Samples were created by the Census Bureau from the 1970 Census. Names, addresses, and detailed geographic information have been removed from the original questionnaire records to avoid identifying individuals. Each complete, national PUS is a representative one-in-a-hundred sample of 1970 households (and vacant units and persons in group quarters); within each selected household, records detailing the characteristics of resident persons (as well as a record for the housing unit itself) are made available.

We purchased a number of Public Use Samples for New Jersey and the New York and Philadelphia metropolitan areas. These included the one-in-a-hundred samples from: the 15% questionnaire for New Jersey counties; the 5% questionnaire for New Jersey counties; the 5% questionnaires for certain parts of Pennsylvania and New York; the 15% questionnaire for New Jersey (without county breakdowns), and several other samples. Of all these data sets, the one most heavily used was the 5% questionnaire for New Jersey counties. The cost of this one data set is $80 from the Census Bureau. A similar data set is available for all states.

Use of a sample from New Jersey, in which small geographical areas can be identified, allowed students to generate and test hypotheses about their local community (since almost all students at Ramapo are from New Jersey). For New Jersey, this one-in-a-hundred, 5% questionnaire contained approximately 95,000 records and included items such as occupation, citizenship, year of immigration (if respondent is an immigrant), in addition to the questions asked of all persons such as race, age, sex, and marital status. Thus, for only $80, it was possible to obtain a large, well-developed, and highly relevant study for use in the classroom. In addition, it is recommended that the user purchase a Data Base Dictionary to be used with the data set. The Data Base Dictionary contains the PUS record formats, variable character-lengths, beginning positions, minimum and maximum values, and labels. This information is contained on a tape file and obviates the need to manually program all of this information in the access program. The cost of a Data Base Dictionary is $70.

Once the data file has been obtained, it is necessary to provide a program to access the data and produce the desired tables. Most university computer centers run packages such as SPSS which may be used to access the data. The census data, however, has a unique property which makes it cumbersome (though not impossible) to use packages like SPSS. The census data are organized as a hierarchical, heterogneous file. (Hierarchical files are defined by regular, sequential patterns among different record types and heterogeneous files contain records of multiple formats.)

In order to process these types of files efficiently, we purchased a package called CENTS-AID, which contains unique features for processing files containing a hierarchical structure and multiple record formats. CENTS-AID was explicitly written for analysis of census data and may be used to produce multi-dimensional cross-tabulations. It was purchased for $600 from DUALabs in Arlington, Virginia. Although this price may seem prohibitive, CENTS-AID saves significant programmer time because it was written to process hierarchical files. It is also considerably cheaper to run on large files than SPSS. Appendix A shows a series of benchmarks run on NJECN's IBM 370/168 comparing processing efficiency between CENTS-AID and SPSS. If $600 still seems like a lot of money, you can always use SPSS.

Use in the Classroom

The census data were used in four sociology courses during the past academic year. These courses were: Sociology--Scope and Methods, an introductory freshman sociology course; Research Practicum--Computer Applications, a freshman- and sophomore-level introduction to sociology research methods; Statistics for the Social Sciences, a junior- and senior-level research course and; Community Structure and Inequality--Methods of Research, a senior-level research seminar.

As can be seen, it was possible to use the census data at all instructional levels. Obviously, the level of expertise required of the student varied among grade level, but nevertheless, all students, even at the most elementary levels, were able to get a feel for the methodology of social science research.

At the freshman level, students use both the pre-printed summary tables and small subsets (approximately 400 cards) of the PUS to answer course assignments. In this course, the students are permitted to work in groups when executing the computer programs. In order to make the course as relevant as possible to the student, they are asked to analyze data for their local communities. First, they are presented with a written assignment that probes their understanding of the economic position, the life styles, and the social and political attitudes of the social classes in their community. The students are expected to formulate hypotheses about these traits in their communities from both their personal experiences and analyses of class structures based upon reading assignments. Then the students are expected to compile tables that selectively draw from relevant material in the 1970 Census from their census tract. Using these tables, they are to compare their own analyses of class structure in their community with the actual 1970 Census data.

In the senior seminar course, Community Structure and Inequality, students use the 95,000 record, 5% Public Use Sample. From these 95,000 records they use the CENTS-AID package to select those relevant to their study. A typical assignment in this course is to create comparative social and economic profiles of two counties. Using the appropriate subsets of the data, the students are asked to present a concise portrait of the labor force, racial, income, household type, and home ownership characteristics of adult residents of the counties they have selected. They are to concentrate on contrasting patterns, the causes of these patterns, and the development of further questions for research.

The assignments in the senior-level course are broader and more open-ended than the assignments in the introductory course. Nevertheless, the census data provide a unique methodological resource which can be tailored to instruction at the elementary levels and still provide enough depth to challenge advanced students.

Conclusions

The use of census data in the sociology curriculum provides a meaningful, well-constructed data set for instructional analysis. It provides both relevant and substantive data which can be used at all levels of instruction. It was found that students enjoyed the opportunity to investigate their local communities and readily absorbed the necessary computer and analytic skills necessary to do so.

The two instructors plan to expand the use of census data analysis into other sociology courses. The existence of the 1960 Public Use Sample and eventually the 1980 Census provides the opportunity to look at longitudinal developments. With this successful introduction, both Ramapo College and NJECN plan to expand their data holdings to provide a large library of data bases for both instruction and research

References

Introduction to CENTS-AID and Computer Analysis of Census and Government Research Data, With Appendices on Using SPSS With the 1970 Census Public Samples. Harold Benenson, published by NJECN, January 1979.

A Student Workbook on the 1970 Census. U.S. Department of Commerce, Bureau of the Census.

The 1970 Census and You - A General Introduction to Census Data. U.S. Department of Commerce, Bureau of the Census.

CENTS-AID II User Manual. DUALabs. Available from National Technical Information Service (NTIS), 5285 Port Royal Road, Springfield, Virginia 22161.

Appendix A

Benchmark Test - CENTS-AID vs. SPSS In Cross-Tabulations From Large Data Files

DUALabs, the creators of CENTS-AID, claim that CENTS-AID is highly efficient for the processing of large conventional data files of simple construction, i.e., containing homogeneous or rectangular arrays of records. They write:

> CENTS-AID employs a COBOL generation logic to process statistical data files... This generative system structure imposes a minimum cost in any tabulation run, while offering the advantages of extremely high efficiency in processing... CENTS-AID II is most efficient and cost-saving when processing large input files. (CENTS-AID User Manual, pg. 3.)

The following Benchmark Test compares the processing efficiency of CENTS-AID vs. SPSS, using a series of increasingly large data files. All files were drawn from the County Group 5% Questionnaire, Public Use Sample (PUS) of Basic Records from the 1970 Census.

The requirements for the test were simple: each program had to generate a cross-tabulation of Occupation (row variable) by Race (column variable) and Sex (control variable), and calculate percentages for all cells and subtotal figures. Only persons who were employed at the time of the 1970 Census enumeration and were sixteen years old or older, were selected for tabulation.

A total of six variables--sex, race, age, detailed occupation, employment status, and PUS record-type indicator--were accessed to select records and generate cross-tabulations.

The size of the input data sets on successive tests were as follows:

Test #1 - 8,310 records (all Hudson County records from PUS 1/100 File No. 506, 5% Questionnaire)
Test #2 - 51,236 records (all records from the six counties which comprise the Newark, Paterson,

and Jersey City Standard Metropolitan Statistical Area's (SMSA's) 1/100 File No. 506, 5%)

Test #3 - 277,590 records (all records in the 1/1000 national 5% County)

Test #4 - 554,972 records (all records from both the 5% and 15% 1/1000 County Group Files Nos. 540 and 240)

CENTS-AID proved to be the more efficient system with very large data files of a quarter or half a million records or more. The differences in CPU times is apparent with 50,000 records (CENTS-AID 14.5 seconds, SPSS 24.9 seconds). However, past the quarter million record mark (277,590), CENTS-AID required only a bit more than one-third of the CPU time used by SPSS (43.4 vs. 123.8 seconds). With more than one-half million records (554,972), the gap grows radically (76.8 vs 263.8 seconds). The difference in CPU time accounts for the relative difference in the costs of these runs.

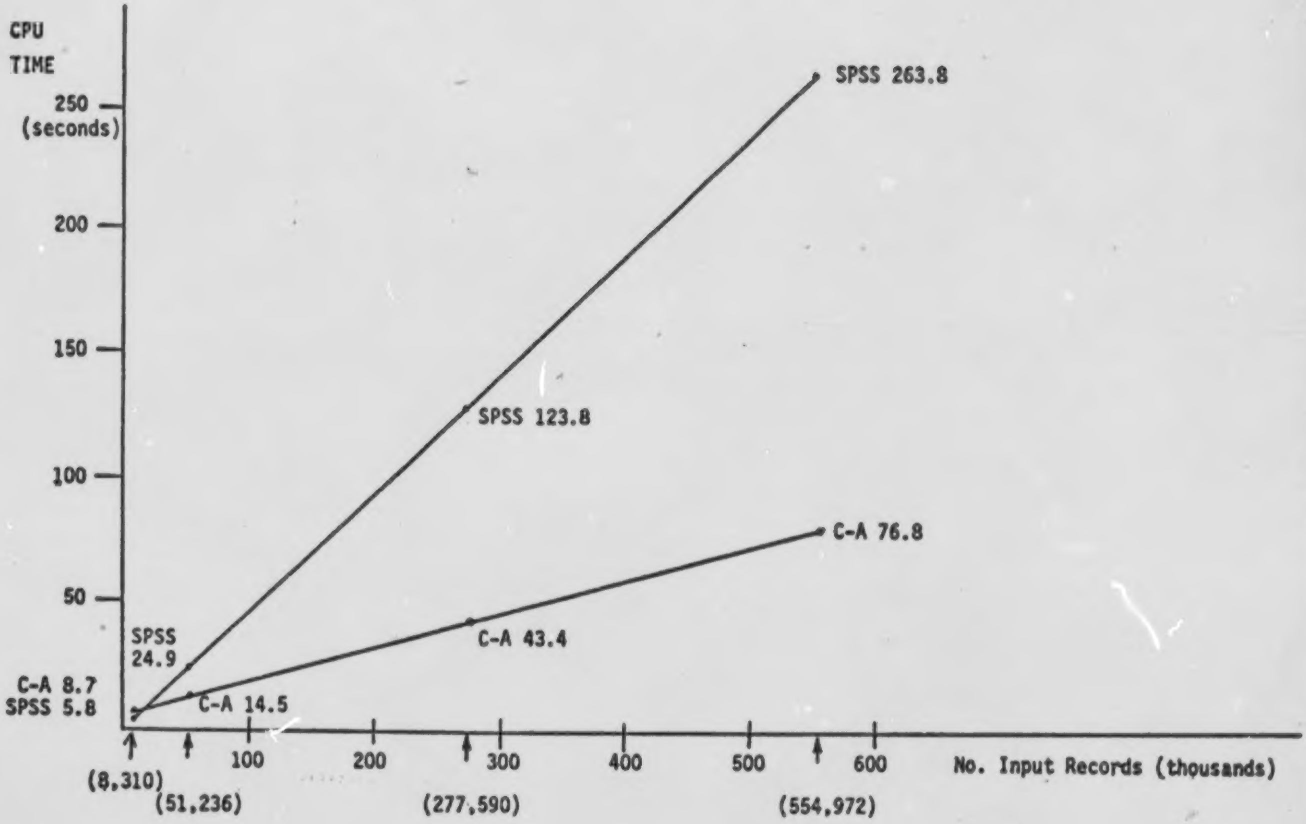
It should be remembered that the Benchmark Test asked for only one table per run. More elaborate requests for tables can be expected to increase the differences observed in CPU times. The differences in CPU times for this test are plotted in Figure 1.

TABLE 1: BENCHMARK TEST RESULTS

	Test #1	Test #2	Test #3	Test #4
No. of Input Records	8,310	51,236	277,590	554,972
Size of Universe	2,563	15,874	76,590	152,104
No. of Variables	6	6	6	6
No. of Tables Generated	1	1	1	1

	Test #1		Test #2		Test #3		Test #4	
	SPSS	CENTS-AID	SPSS	CENTS-AID	SPSS	CENTS-AID	SPSS	CENTS-AID
Core Storage	282	200	282	200	282	200	282	200
CPU Time (seconds)	5.8	8.7	24.9	14.5	123.8	43.4	263.8	76.8
I/O Time (seconds)	4	16.0	13	23	60	59	118	105
Dollar Cost	2.02	3.37	6.86	5.00	31.71	12.63	67.11	22.16

FIGURE 1


CPU
TIME
(seconds)
250
200
150
100
50
SPSS 263.8
SPSS 123.8
C-A 76.8
C-A 43.4
SPSS
24.9
C-A 8.7
SPSS 5.8
C-A 14.5
100
200
300
400
500
600
No. Input Records (thousands)
(8,310)
(51,236)
(277,590)
(554,972)

COMPUTER-ASSISTED INSTRUCTION FOR LARGE CLASSES IN ECONOMICS

Donald W. Paden and Michael D. Barr
University of Illinois

During the Fall 1977 semester, roughly 400 students in two lecture sections of Principles of Economics at the University of Illinois, Champaign-Urbana, were exposed to instruction involving the computer. This paper reports on the achievement of these two groups, their attitudes towards the course, the cost of instruction, and the amount of time students spent on the course both in and out of class.

The material on the computer to which the students were exposed consisted of three components: a series of seven review and expository lessons, each taking from 30 to 50 minutes for students to complete; three multiple-choice hour examinations each with multiple forms; and a record-keeping system for tallying such information as the time spent by students on the system, numerical scores on the lessons, and scores and letter grades on the hour examinations. Other pencil and paper grades were entered on the system to facilitate the computation of the course grade (i.e., quizzes, midterm, and final examination). These records were available for student inspection at any time. The package was delivered by PLATO[1]—the University of Illinois' version of Computer-Assisted Instruction. One group of students was given only the hour examinations administered by PLATO. The other group of students was required to complete the seven lessons as well as the hour examinations on PLATO. The record-keeping system was used for both groups.

PLATO consists of a display panel similar to a television screen, a typewriter-like keyset, and a computer. Students communicate with the computer by using the keyset and the computer presents material to students, asks them questions, judges their answers and responds to their errors on the display panel, and stores information about their responses. As many students can work simultaneously on a single lesson (or lessons) as there are terminals. Responses from the computer are displayed almost instantaneously for the student to read and respond to. The system provides students working on lessons with a measure of animation in visual material impossible with the printed page and gives them some control over graphs and charts. An index for each lesson permits skipping around within lessons. A general index is available for selecting other options such as review sequences, announcements, asking questions, seeing grade distributions, and writing notes to the instructor. The student is also given access to the computer for making computations and, upon occasion, for solving equations with values supplied by the student. Thus, PLATO is used to insure the interaction of students with lesson material and, in a more general sense, provides another option to enhance the study of economics (along with conventional texts, programmed material, lectures, quiz sessions, films, workbooks, tutors, and television). In addition, the computer serves as a servomechanism in monitoring and recording student performance and as a medium for communication between students and instructors.

Design of the Experiment

One large group of students took only the hour examinations on PLATO while attending two lectures and two quiz sessions each week. This was designated as the non-PLATO or control group. The other large group of students also attended two quiz sessions but went to only one lecture each week—substituting roughly an hour on PLATO for an hour of lecture. This constituted the PLATO or experimental group. There were ten quiz sections taught by five teaching assistants. Roughly half of the students in the second group were intermingled in quiz sections with students from the first group and thus were exposed to the same teaching assistants. The lecturer, textbook assignments, and outside readings were the same for all students. The remainder of the students in the second group was handled in quiz sections taught by the lecturer and the PLATO programmer. These were labelled other-PLATO students and, although exposed to the same instruction as the PLATO students, did not share common quiz instructors with the control group. Although the students were not assigned randomly to the two treatments, there is no obvious reason to suspect bias.

Performance on the Final Examination

The results of the final examination are shown in Table 1, where students are classified by the three groups indicated above.

The mean score on the final examination for students who studied content material on PLATO was significantly higher than for the non-PLATO students. The mean score of the other-PLATO students (who worked the PLATO lessons but did not share common teaching assistants with the control group)

Table 1

Performance of PLATO and non-PLATO Students on the Final Examination

	n	Mean Scores	Differences	Standard Deviation
Non-PLATO (control)	206	52.08	1.20*	7.06
PLATO (experimental)	124	53.28	.41	5.81
Other-PLATO	106	52.49		7.13

*Significant at the 10 percent level when compared with the control group.

was also higher than the non-PLATO students, although the difference was not statistically significant. These results agree generally with three other studies in which the authors have participated—reported upon elsewhere by Paden, Dalgaard, and Barr. In these studies PLATO students have also generally out-performed non-PLATO students, but not always by a statistically significant amount—at least as measured by scores on the final examination. In light of this generally favorable background, the evidence provided by the present experiment inclines one to believe that for this one-semester micro-macro introductory course, PLATO made a difference. Remembering that computer-assisted instruction was substituted for only one of the four contact hours normally involved in the course and only one out of a total of seven hours spent on the course each week by the average student, the improvement looks not only statistically significant, but quantitatively important. In evaluating this gain it should be remembered that three of the ten hours spent on PLATO in the present study consisted of answering common examination questions, rather than working on content instruction.

Time Spent on the Course

At the time of the final examination, students were asked the amount of time they spent studying economics, excluding class time or time on PLATO. Averages for the three groups are shown in Table 2.

Taking the figures at face value and accumulating the saving for the semester, the two PLATO groups responded that they spent almost six and seven hours less time, respectively, on the course during the semester than the non-PLATO group—probably as much time as most students spend studying for the final examination in economics.[2] Not only did PLATO students get better grades in economics, but there obviously was a trade-off between higher grades in their other courses and more leisure, depending on how they used the time saved.

Of some methodological interest is another attempt to arrive at the time spent on the course. Instead of asking students directly how much time they spent studying economics, they were told that on the average their classmates spent roughly three hours taking examinations on PLATO. They were then asked to check, using the categories in Table 3, how the time they spent on the examinations compared with this average. Since the computer actually recorded the exact time each student spent on the examinations, it was possible to compute the averages shown in Table 3 for students who identified themselves as being in each group.

Next, students were told that on the average, their classmates spent roughly three hours per week studying economics, not including time spent in class or on PLATO. Again they were asked to check how this general average compared with their own study time outside of class—using the same five categories indicated in Table 3. After tabulating the information on the number of students in each

Table 2

Time Spent Studying Economics per Week, Excluding Class Time and Time on PLATO

	n	Means	Standard Deviations
Non-PLATO (control)	222	3.72 hours	1.88 hours
PLATO (experimental)	104	3.30 hours*	1.65 hours
Other-PLATO	100	3.26 hours**	1.53 hours

*Significant at the 10 percent level when compared with the control group.

**Significant at the 5 percent level when compared with the control group.

Table 3

Estimated Time and Actual Mean Time Spent on Examinations by PLATO and non-PLATO Students

		Actual Time	
Estimated Time	Non-PLATO	PLATO	Other-PLATO
Considerably more than 3 hours	5.42 hours	4.63 hours	3.92 hours
Somewhat more than 3 hours	4.09 hours	4.68 hours	3.84 hours
About three hours	3.51 hours	3.13 hours	3.15 hours
Somewhat less than 3 hours	2.87 hours	2.71 hours	2.40 hours
Considerably less than 3 hours	2.94 hours	2.95 hours	1.68 hours

category (Appendix 1) and using the time dimensions shown in Table 3 for considerably more, somewhat more, etc., an average amount of time was computed for each of the three groups of students. As was true when the students were asked directly, the average amount of time spent studying economics outside class was greatest for the non-PLATO group, next largest for the PLATO group and least for the other-PLATO students (3.61, 3.50 and 3.16 hours, respectively).[3] Unfortunately, the three hours specified for both the above questions was too low for all three groups and may have introduced a bias in the computed means. Under the circumstances the times calculated from the direct responses of students (Table 2) are thought to be the more meaningful of the two sets of estimates.

Quite apart from their use in making the above estimates, the figures in Table 3 reveal surprising differences in the average time spent taking the PLATO examinations, ranging from roughly 3 to 5 1/2 hours for non-PLATO students to 1 1/2 to 4 hours for the other-PLATO students. Individual students undoubtedly differed much more than this. As an example, one A student spent only six hours on both the lessons and examinations combined. An examination of the questionnaires showing the time students spent studying economics outside of class revealed differences ranging from one to 14 hours per week.

Student Attitudes

During the latter part of the semester a University of Illinois Course Evaluation Questionnaire was distributed to students during a lecture period. Responses to three relevant questions are shown in Table 4.

A casual inspection of the figures indicates that students using PLATO tended to react more favorably to the course in general, to the course content, and to the lecturer than students who did not.[4] The responses to the questionnaires were also tabulated by anticipated grade. The results for the course in general are shown in Table 5. Again, from an inspection of the percentages in Table 5 it is apparent that when the responses are classified by anticipated grade, A and B students who had used PLATO were more favorably inclined toward the course than A and B students who did not. Although a sizeable proportion of the students received grades of C or less, the reluctance of students to admit this on the questionnaire before the end of the course resulted in too few students in this category. The interpretation of the figures for C students, therefore,

Table 4

Student Evaluation of the Course and the Lecturer by non-PLATO and PLATO Students (percent responding)

	Course in General		Course Content		Lecturer	
	Non-PLATO	PLATO*	Non-PLATO	PLATO*	Non-PLATO	PLATO*
Excellent	4	16 (10)	5	21 (18)	13	16 (12)
Good	39	45 (47)	48	46 (45)	36	45 (46)
Satisfactory	39	24 (24)	35	24 (25)	30	24 (28)
Adequate	11	10 (10)	8	8 (9)	9	10 (7)
Poor or worse	7 100	5 (4) 100	4 100	1 (3) 100	12 100	5 (8) 100

*Figures in parentheses are the other-PLATO students.

Table 5

Student Evaluation for the Course in General by Anticipated Grade
(percent responding)

	A Students		B Students		C or less	
	Non-PLATO	PLATO*	Non-PLATO	PLATO*	Non-PLATO	PLATO*
Excellent or good	53	78 (79)	43	58 (49)	50	29 (86)
Satisfactory	32	19 (14)	47	30 (34)	33	43 (14)
Adequate or poor	15	4 (7)	10	12 (17)	17	28 —
	100	100	100	100	100	100

*Figures in parentheses are for other-PLATO students.

is open to question. Although the table shows that superior students seemed to be more enthusiastic about PLATO than other students, the question of who benefited the most from using PLATO remains unanswered.

Other Reactions to PLATO

Student reaction to the use of PLATO was explored further through the items listed in Table 6.

On the basis of the answers to Questions 1 and 2 PLATO students seemed to be satisfied with the quality of the lessons and the contributions the lessons made to the course. The PLATO lessons viewed as a supplement to the course were overwhelmingly endorsed by PLATO users (Question 4). Again, as judged by the last question, PLATO students seemed satisfied with their exposure to computer-assisted instruction while non-PLATO students would have opted for more. These same results have been true for other semesters and for other courses. Although the preference for PLATO could be due to the novelty of computer-assisted instruction, this is thought to be somewhat unlikely due to its widespread use on the University of Illinois campus. In interpreting these figures it should be remembered that *all*

Table 6

Student Reaction to PLATO
(percent responding)

1. How much did PLATO contribute to course?		Very Much		Non-committal		Very Little
	Non-PLATO	9	29	20	20	20
	PLATO	34	44	16	4	2
2. The overall quality of the PLATO material was:		Very High		Non-committal		Very Low
	Non-PLATO	10	30	43	8	9
	PLATO	26	54	16	3	1
3. Learning from PLATO in the course was:		Enjoyable		Non-committal		Unenjoyable
	Non-PLATO	22	30	31	11	8
	PLATO	34	36	16	9	3
4. PLATO as a supplement to this course was:		Helpful		Non-committal		Of No Help
	Non-PLATO	18	33	27	14	9
	PLATO	44	43	6	7	0
5. Compared to learning by textbook, PLATO was:		Faster		Same		Slower
	Non-PLATO	16	26	49	9	3
	PLATO	35	36	25	4	0
6. Compared to learning by lectures, PLATO was:		Faster		Same		Slower
	Non-PLATO	18	28	47	6	3
	PLATO	27	39	28	6	0
7. The number of exercises on PLATO was:		Too Many		About Right		Too Few
	Non-PLATO	1	3	60	18	18
	PLATO	1	16	69	10	4

of the students in the course used PLATO at least for taking examinations. In stating their opinions, therefore, non-PLATO students were doing so on the basis of more than a minimum exposure to computer-assisted instruction.

Cost of Instruction

The estimated cost of instruction in the economics course in question (without PLATO) is shown in the first column of Table 7. The Computer-Based Education Research Laboratory has suggested a ball-park figure of $2.50 per student hour as representing the cost of PLATO instruction (Column 2). This includes such items as computer use, local communications, and depreciation on the terminal but does not include the cost of lesson development.[5] The additional cost of PLATO per hour is shown in the third column. Thus, if PLATO is used as a substitute for one lecture per week for 15 weeks, the additional cost is roughly $26 per student per semester; if substituted for one quiz session taught by a graduate teaching assistant, the additional cost is about $22. For a possible two percent gain in the average score of students on the final examination—plus improvement in student attitudes--this may seem unduly costly. On the other hand, if the six-hour saving of student study time per semester is reasonably accurate and if a value of $4 an hour is placed on the long-run opportunity cost of this time (possibly an underestimate), there is a small net gain from using PLATO as a substitute for teaching assistants in small classes. Given the nature of the above estimates and the likelihood of PLATO costs declining in the future, the figures are offered only as a matter of general interest.

Conclusions

The present study provides reasonably impressive evidence that students in a controlled situation who depended upon PLATO for roughly one-seventh of the input in the beginning course in economics scored higher on the final examination than those who did not use PLATO. Although the gain was certainly not spectacular, in the opinion of the authors the improvement was quantitatively important. In a somewhat less elegantly executed part of the study the evidence suggests that students who used PLATO spent less time on the course than other students. When this was taken into account, the opportunity cost of PLATO instruction compared favorably with that of traditional instruction. Finally, the study indicates that students who used PLATO felt better about the course than those who do not.

The gains in learning and attitudes suggested above may be due to the care used in writing the seven instructional sequences or to the fact that computer-assisted instruction, via PLATO, required students to maintain a cognitive posture during most of the time they spent on the system. More subtle gains unique to the PLATO experience were more difficult to measure. On the one hand, the economics programs on PLATO encouraged an appreciation of the use of models, mathematics, and the computer. On the other hand, PLATO made possible tutorial sessions free of judgmental overtones which, for many students, tends to inhibit learning.

The authors believe that these same quantitative and qualitative gains will be found to be true in other institutions, for instructors using similar textbooks, and teaching comparable courses.[6]

Table 7

Costs of Instruction

	Classroom Instruction	PLATO Instruction	Additional Expense (per hour)	Additional Expense (per semester)
Lecture	$.75/hr	$2.50	$1.75	$26.25
Quiz Sections	1.00/hr	2.50	1.50	22.50

Notes

1. PLATO is an acronym for "programmed logic for automated teaching operations".
2. In comparing the number of students in these groups with those in Table 1, it is apparent that there was some non-response and some group misidentification which could not be tracked down.
3. The time spent by students attending the two lectures (for the non-PLATO group) and the sum of the time spent attending one lecture and working on the computer (for the two PLATO groups) is thought to be roughly the same. Almost no student failed to finish the PLATO lessons. The non-PLATO students were coerced into attending at least one lecture each week by giving credit for attendance (balancing the credit given the PLATO students for working on the lessons). Although attendance was not taken in the quiz sections or in the other lecture, there is no reason to believe that absences were more numerous for one group than the others.
4. Although student replies to the questionnaire were anonymous, they reflect only the opinions of those who attended the lectures on the day the forms were distributed to the two groups.
5. Each lesson probably required from 100 to 300 hours of preparation on the part of the authors and programmers. James H. Wilson and M. Eugene Moyer helped program some of the earlier material.
6. The lessons are currently being used in such widely differing institutions and programs as community colleges and home economics. They are suitable for both three- and six-hour

courses and for review in intermediate micro and macro courses.

Reference

Paden, D. W.; Dalgaard, B. R.; and Barr, M. D., "A Decade of Computer-Assisted Instruction," Journal of Economic Education, Fall 1977, Vol. 9, No. 1, 14-20.

Appendix 1

Number of Students by Estimated Time Spent Studying Economics Excluding Time in Class and on PLATO

	Non-PLATO	PLATO	Other-PLATO
Considerably more than 3 hours	26	16	10
Somewhat more than 3 hours	40	23	27
About 3 hours	62	30	38
Somewhat less than 3 hours	63	26	22
Considerably less than 3 hours	21	10	6

TESSELLATIONS
AND THE
TEKTRONIX 4051

Elliot A. Tanis
Hope College
Holland MI 49423
(616) 392-5111

1. Introduction

The Dutch artist M.C. Escher was intrigued by the idea of filling the plane with figures of animals or people without overlapping or gaps. In turn mathematicians are intrigued by many of Escher's works and have studied them to determine the geometrical basis for his drawings. After the mathematics is understood, a fascinating problem is to have the computer reproduce a similar type drawing.

2. Tessellations

A tessellation or tiling of the plane is a covering of the plane by one or more figures, repeating the figures, and having no gaps and no overlapping of the figures. There are many different types of tessellations and these are discussed in several books and journals. The references list four of these sources [1,2,3,4]. In this paper we shall give one example based on a tessellation of rectangles and the other based on a tessellation of regular hexagons. The examples are very similar to works by M.C. Escher that appear in several publications, one of which is listed in the references [1].

3. Pegasus

A tessellation of rectangles is not the most interesting although it is very common, for example, cement block walls and tiled floors. One of M.C. Escher's works that is fairly easy to analyze is based on a tessellation of deformed rectangles.

To construct Pegasus, the left side of a rectangle was deformed. This deformed side was then translated horizontally to deform the right side in the same way. The top side of the rectangle was deformed and the deformed side was then translated vertically to deform the lower side. The addition of an eye turns the deformed rectangle into Pegasus. The key idea for this construction is contained in an excellent book by Ranucci and Teeters [3].

In Figure 1 the original and deformed rectangle are depicted. The coordinates of the lower left and upper right vertices of the rectangle are (2,2) and (7,6), respectively.

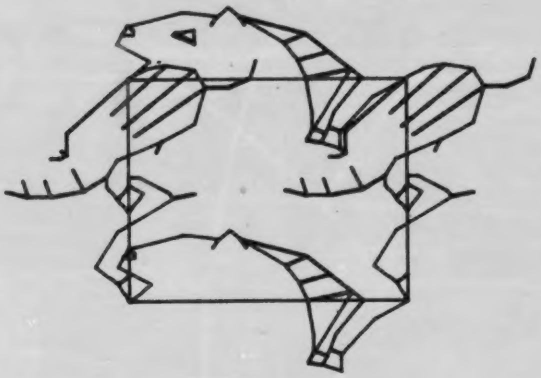


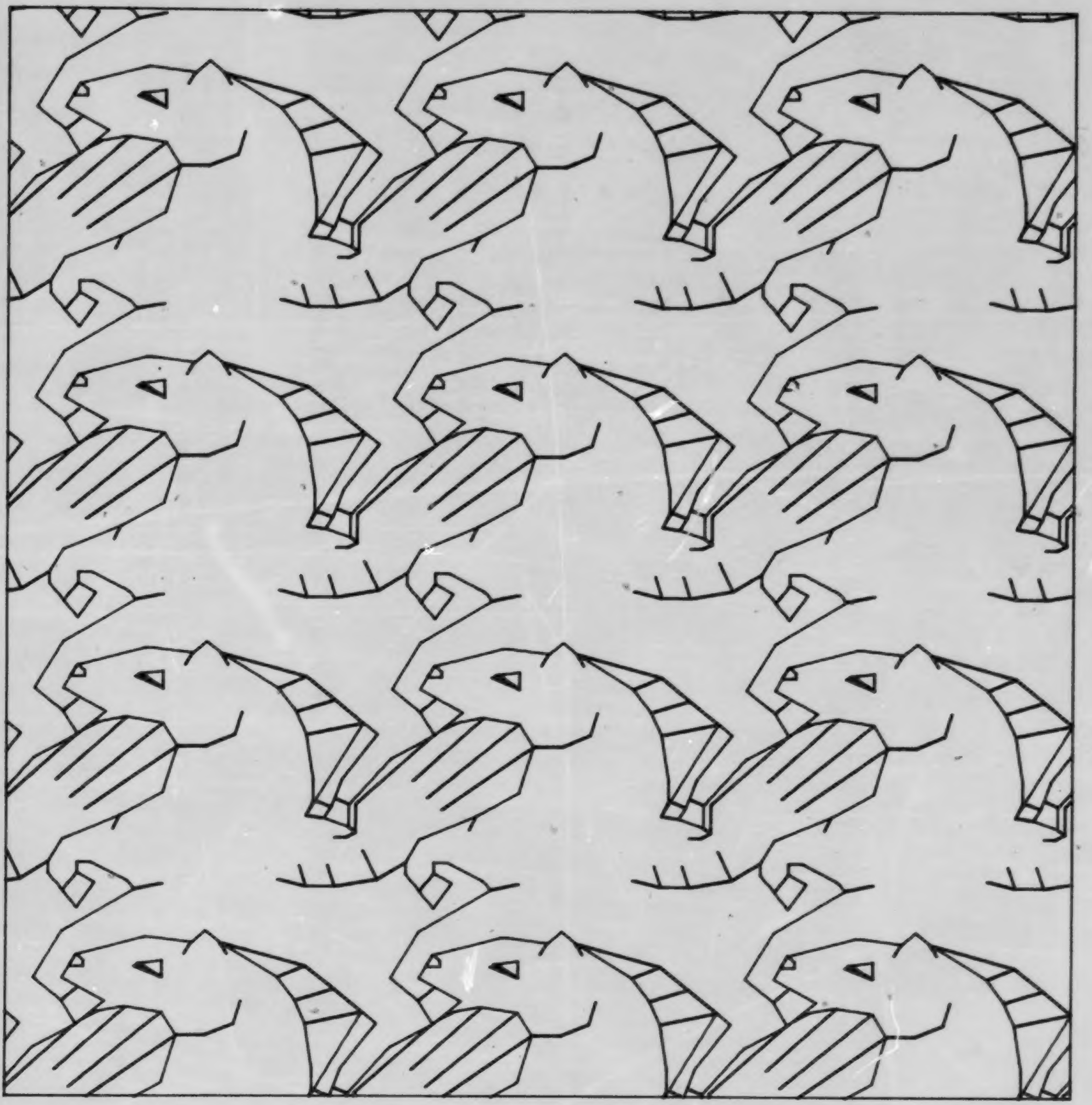
FIGURE 1

Because of the way in which Pegasus was constructed, this figure will tessellate the plane. Each of the vertical and horizontal arrays is simply translated and drawn repeatedly to fill the desired planar region. An example is given in Figure 2.

4. Reptiles

Another regular polygonal tessellation is that based on regular hexagons. The hexagon can be deformed in different ways to yield an interesting tessellation. For example, three adjacent sides can be deformed and translated across the hexagon to deform the opposite sides. A turkey tessellation constructed in this manner is given in *The Mathematics Student* [5].

We deformed the hexagon in a different



239

FIGURE 2

way. The lower side of the hexagon was deformed and then rotated -120° about its right end point to deform side 2. Proceeding counterclockwise, the third side was deformed and rotated -120° about an end to deform the fourth side. Similarly the fifth side was deformed and rotated -120° about an end to deform the sixth side. Two interior arrays were added to yield the reptile depicted in Figure 3. Much credit for this figure must again be given to Ranucci and Teeters [3].

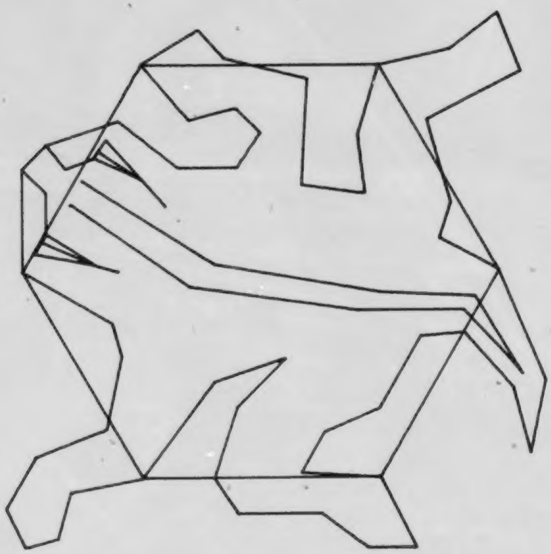


FIGURE 3

After one figure has been developed, it is a challenge to have the computer tessellate a planar region without drawing any parts twice. Note that those three vertices about which the three deformed sides of the original hexagon were rotated become points of three-fold symmetry in the tessellation. Using the ROTATE and RDRAW commands in the Tektronix BASIC, it becomes a matter of moving to these key points and drawing the proper array three times -- the original array, and the array rotated 120° and 240°.

An example of a tessellation of the reptiles is depicted in Figure 4. In the terminology of the plane symmetry groups, this tessellation would be classified as a p3. The article by Doris Schattschneider gives an excellent account of plane symmetry groups [4].

5. Applications in the curriculum

The two examples illustrate how the computer and the works of M.C. Escher can be used to illustrate translations and rotations in the plane. These examples are excellent for a college or high school geometry class. Examples of reflections and other tessellations have also been developed and could be used to help the students understand geometrical transformations.

The programs that were written for the Tektronix 4051 to make these drawings were designed so that a student may input his or her own data. The resulting tessellation is sometimes quite interesting. Given a tessellation, the challenge to the student is to see some interesting creature or other form in the resulting design.

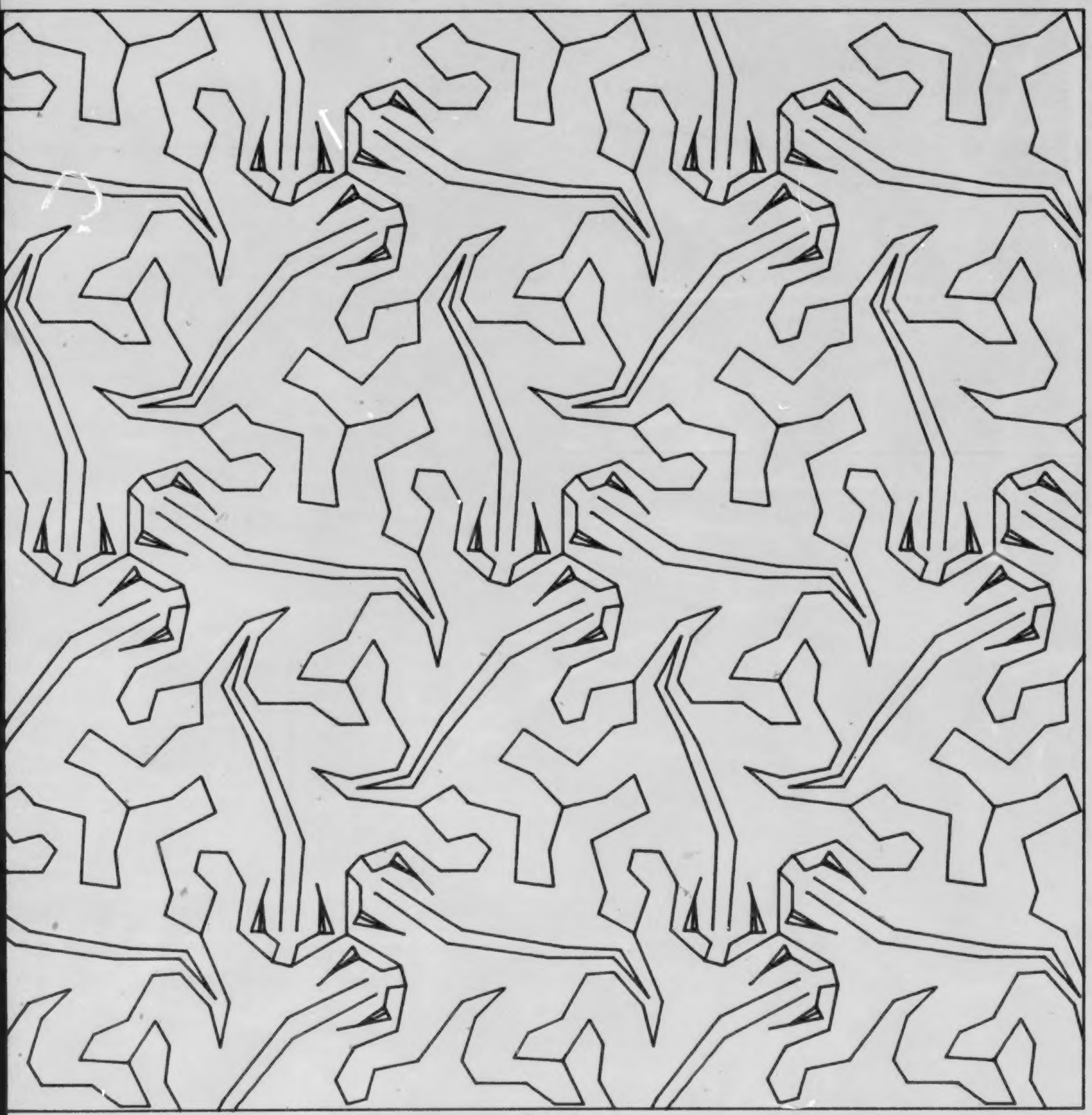
These examples also illustrate how the computer and tessellations can form the backbone for an independent study project. Two students, one of whom plans to become a high school mathematics teacher, selected such projects for the 1978-79 academic year. These students have gained a greater appreciation of various types of tessellations by writing computer programs to draw the tessellations.

6. Distribution of materials

The author is willing to share these two programs if you would like to use them on your Tektronix 4051. If you are interested, send a blank tape along with sufficient funds to cover return postage. We will copy the programs for you.

References

1. Caroline H. MacGillavry, Fantasy & Symmetry, The Periodic Drawings of M.C. Escher, Harry Abrams, New York, 1976.
2. Phares G. O'Daffer and Stanley R. Clemens, Geometry: An Investigative Approach, Addison-Wesley, 1976, pp. 75-153.
3. E.R. Ranucci and J.L. Teeters, Creating Escher-Type Drawings, Creative Publications, 1977.
4. Doris Schattschneider, The plane symmetry groups: Their recognition and notation, The American Mathematical Monthly, Vol. 85, No. 6 (1978) pp. 439-450.
5. The Mathematics Student, Vol. 26, No. 2, (1978).



241

FIGURE 4

COMPUTER GRAPHICS AS A LECTURE AID IN UNDERGRADUATE MATHEMATICS

Gerald J. Porter *
University of Pennsylvania

*Research supported by National Science Foundation Grant SED76-20654.

For the past year we have been using computer-generated graphics as a teaching aid in undergraduate mathematics courses at the University of Pennsylvania. It is our goal in this report to discuss our usage attempting to spotlight those topics most suited for computer graphic assistance. In addition we comment upon on-line vs. prerecorded displays.

Courses Involved

The courses in which the graphic displays were used were the standard first-year calculus course, a second-year calculus course intended for mathematics majors, and an upper class course on differential equations. None of these courses involved any student use of the computer, nor was it assumed that the students had any knowledge of the computer. The first-year course had an enrollment varying between 100 and 150. The other courses had enrollments of less than twenty students.

Equipment Used

The first-year calculus course meets in an amphitheatre-type lecture room. For our computer-generated graphics displays we use a CRT graphics terminal (Princeton Electronic Products 801). The video output is fed into a TV projector (Kalart-Victor Telebeam) and projected upon a standard movie screen. The terminal is connected with an IBM 370/168 via a 300 baud dial-up line. Programs are written in APL. The screen is clearly visible if only the front lights in the room are turned off. This permits the students to take notes while seeing the displays.

The small classes met in a classroom which seats 24. At first only the CRT was used. A 23" TV monitor has now been mounted in the front corner of the room. Together these two displays adequately cover the room. The classroom lights are shut off and blinds closed when the displays are used. As in the large lecture room this terminal is connected with an IBM 370/168 via a 300 baud dial-up line.

Examples of Applications

For a computer-generated graphic display to be successful it must illustrate some concept in a way that is superior to the usual chalkboard demonstration. The topics which we have found lend themselves to such a treatment are those which require a degree of precision which is not usually possible with a freehand sketch and those which have an important dynamic aspect such as convergence or solution of differential equations.

To emphasize this point we discuss in detail some of the applications which we consider to have been successful. It is not our goal to give an exhaustive list of such applications. Rather we present these so that the reader can note those characteristics which can best be illustrated by computer-generated graphic displays. We conjecture that worthwhile applications of graphics in other fields share many of the same characteristics.

The convergence of a sequence of functions is a topic which requires precision in comparing the n^{th} function with the $(n+1)^{st}$ function. By using the computer one can actually see the partial sums of a Taylor series approaching the function. The dynamic aspect of the display is important in watching the partial sum stick to the function up to a certain point and then change. Divergence can also be illustrated in this manner. For example, if one studies the Taylor series for $f(x) = (1+x)^{-1}$, then one can see the partial sums converging to $f(x)$ for $|x| < 1$ and getting further and further from $f(x)$ for $|x| > 1$.

In a similar fashion convergence of the partial sums of a Fourier series is a topic which is virtually impossible to illustrate accurately on a chalkboard. Yet by using the computer one can easily show that convergence. An interesting aspect of Fourier series convergence is the so-called Gibbs' effect. The phenomenon is clearly visible in the computer-generated graphs of the partial sums.

Newton's method for computing the zeros of a function is another iterative process which lends itself to a dynamic display. The usual picture shows the procedure of taking the tangent to the curve $f(x)$ at the point $(a_n, f(a_n))$ and setting a_{n+1} equal to the point of intersection of this line with the x-axis. One often illustrates this procedure by finding the square root of two (i.e. a zero of $f(x) = x^2 - 2$) . In this case if one chooses a_0 to be any number greater than $\sqrt{2}$,the convergence is monotonic and rapid. Interesting examples of nonmonotonic and divergent behavior can easily be seen. Nonmonotonicity is illustrated by choosing a function such as $f(x) = x - \cos x$ and letting a_0 be large (eg.35). Since $f'(x) = 0$ whenever $\sin x = -1$ there are many points near which convergence is bad. Non-convergence can be illustrated by $f(x) = x^2+1$. In this case one can see Newton's method trying its best to converge but, of course, not succeeding. An example which we accidentally discovered is a cubic equation for which three very close initial approximations led to three distinct roots. Such examples lead to interesting questions which can be further explored by using the computer.

It is easy to give an example of a continuous function which is not differentiable at a point $(f(x) = |x|)$ or even at several points. Weierstrass gave an example of an everywhere continuous function which is nowhere differentiable. The very nature of this function makes it difficult to study. Computer-generated graphics can be used to examine this function by successively zooming in on the graph in a neighborhood of a point. While it is true that for any particular picture the function illustrated is still differentiable almost everywhere, it is possible to gain a deeper insight into the nature of the example by using the computer.

Many equations of motion are given parametrically in the plane, and parametric equations are an important topic in introductory calculus. Yet it is often impossible to draw the paths given parametrically except for cycloids, epicycloids, etc. and those equations which can be converted easily into cartesian coordinates. Using computer-generated graphics, a far greater variety of such examples can be studied. The Lissajous functions $((x,y) = (\sin t, \sin k(t+b)))$ are good examples for further study.

In the same way a greater number of graphs of functions given in polar coordinates can be studied. In both this case and the previous one, care must be taken to use the computer to deepen our knowledge of the mathematics rather than as a crutch to avoid thinking.

Curve sketching is a topic for which computer-generated graphics are inappropriate. Computer-generated graphs are precise whereas the idea of using the derivative to sketch the graph involves using only imprecise information (e.g. the function is increasing at x). Nevertheless it is an instructive exercise to view the graph of f(x) directly above that of f'(x) . In this way one can directly see that f'(x) > 0 means f'(x) increasing; f"(x) < 0 at an extreme point means a local maximum; and so forth. In fact after viewing such displays we have added a new type of exercise to our course: Given the graph of f'(x), sketch some f(x). This type of problem helps teach the relationship between f'(x) and f(x) in a visual rather than analytic way. In addition one can understand what various types of discontinuties in the derivative imply for the function. It is not easy to show these phenomena in terms of the usual curve sketching problems.

Some elementary differential equations are often included in the first-year calculus course. The equation y" = g , describing free fall, is a standard example. This topic is often difficult to teach since many students have studied this topic in high school physics and have memorized the various formulae. A more interesting example (in connection with parametric equations) is that of two-dimensional motion of a thrown (shot) object. We have used the example of shooting a basketball to illustrate this topic. The magnitude and the direction of the initial velocity must be chosen so that the ball passes through the basket. After various experiments are tried, the problem is solved analytically.

The dynamic aspects of differential equations (time is often the independent variable) together with the precision needed to draw the solution curves make computer-generated graphics especially valuable in any course dealing with this subject.

The second order linear equation, $y''+ ay'+ by = k \cos t$, $(a,b \geq 0)$ is fundamental in describing the behavior of mechanical and electrical systems. For the homogeneous system (k=0) the various types of solutions (depending upon a and b) overdamped, critically damped, and simple harmonic motion, are all nicely illustrated by the computer. In the nonhomogeneous case (forced vibrations) one can see the effect of resonance. More important, one can see in a dramatic way the manner in which the transient solution dies away and the steady state solution takes over.

Another topic that is better understood using computer-generated graphics is the study of a pair of coupled first order differential equations. These may be interpreted either as a dynamical system or as a phase plane portrait. One nice aspect of the presentation is that if the system is viewed as a dynamical system, then the speed at which the trajectories are drawn is proportional to the speed along the given solution curve. Among the phenomena which can be studied in these examples are the classification of equilibrium points. If the equations are linear one can see the relationship between the eigenvalues and the type of equilibrium, and the role of the eigenvectors is equally well illustrated.

Non-linear equations of Voltera-Lotka type are of value both in creating mathematical models and in instructing students how to sketch trajectories and classify equilibrium points. These systems may be used, together with computer-generated graphics, as examples of deterministic models in a simulation course or a course on mathematical models. The graphics make the results more comprehensible than lists of numbers (i.e. the numerical values of solutions).

Although we have not taught numerical methods for solving differential equations, the task of producing these curves on the graphics terminal is a good motivation for introducing this topic. One can then study visually error buildup and the difference in error between various methods.

An important theorem in the study of the Sturm-Liouville equation is that if q(x) is a positive function and

$y_\lambda(x)$ is a solution to the differential equation

$$y'' + \lambda q(x)y = 0$$

then as $\lambda \to \infty$ the number of zeros of $y_\lambda(x)$ in $[a,b]$ goes to infinity. This proof is again dynamical in nature. Using computer graphics and viewing increasing values of λ one can actually see the zeros being pushed into the interval and obtain a visual understanding of the proof of the theorem.

As a final example we note that it is usually difficult to sketch pictures of surfaces. In part this is due to the intrinsic nature of the problem, reducing three-dimensional information to two dimensions. This is a topic where the computer can be of help. Several factors must be taken into consideration: first, the functions needed to draw two-dimensional pictures of three-dimensional objects are non-trivial to write. This is due to the problem of removing hidden lines. The graphs drawn without hidden lines being removed are often (but not always) useless. Even when one is presented with a working program care must be taken to choose the viewpoint, the window, and other parameters so as to make the resulting picture helpful in visualizing the surface. If this is not done, the computer-generated picture will increase the student's confusion rather than help eliminate it.

Other topics which are amenable to study using computer-generated graphics include two-dimensional linear transformation and the related planar symmetry groups, partial differential equations, and conformal mappings.

Some topics which appear to lend themselves to exposition via computer-generated graphics actually gain little over the more traditional approaches. For example the tangent line to a curve can be defined as the limit of secant lines, and limiting process can be illustrated by the computer. The precision needed for this presentation to be effective is low, and the chalkboard is entirely adequate for teaching this concept. Much the same is true in the use of upper and lower sums to define the integral. The computer is useful in the calculation of the upper and lower sums, but the sketches of these sums is no less successful than the more precise computer-drawn pictures.

On-line vs. Off-line

In 1967 the National Science Foundation sponsored a conference on Computers in Undergraduate Education at the University of Maryland. In calling for experimentation with computer-connected display devices they stated:

> Computer connected display devices, when used in the classroom, can provide the instructor with a means for illustrating sophisticated mathematical concepts graphically in such a way as to allow for almost instantaneous response to changes in input that might be suggested by the students. Indeed, such a capability in the classroom might elicit more student reaction than is generally possible by standard classroom techniques, even when augmented by canned films.

All of our computer usage has been real time on-line. In a lecture of a hundred or more students the instructor must constantly keep the students' attention. Graphic presentations must be carefully prepared. In class communications with the computer must be limited to no more than entering a formula or a parameter. There is no time to go back and correct a programming error or be creative and write a new program. Even so there is a limited ability to react to changes in input that might be suggested by the students. Our experience has led us to believe that it is only the truly exceptional student who will make any suggestion. The vast majority of the students are struggling to absorb the material, and their level of understanding is inadequate to enable them to interact with the display. This inability to provide feedback is also true in the smaller classes. Our colleagues, who already have a deep understanding of the material, almost always are ready to ask, "What if...."; the students are not. Students generally are not knowledgeable enough to ask interesting questions. Perhaps a small class could be trained to react to displays with good exploratory questions; it is our conclusion that this is impossible in large lectures.

What are the implications of this? To begin with we feel that for the most part videotape is an adequate medium for the presentations of the type described above. Videotape has several advantages. Being off-line one does not have to worry about the computer going down. No interaction time is required in class except turning the recorder on. All preparation can be done and the results viewed before class begins. Many classroom instructors who are not knowledgeable about computers and as a result would not use live displays might be willing to use videotapes.

Why not then use some of the excellent films that have been prepared or why not simply distribute videotapes? For some applications this would be ideal. The instructor's flexibility is severely limited if she/he does not have the freedom to try out new approaches, emphasize those concepts which she/he thinks are most important, and tailor the material to the needs of a particular class. Although the class may not ask "What happens if...", the instructor can and often will get a new insight on an old problem or see a new approach to teaching a concept. These new understandings would be impossible in a fixed format type of presentation. For these reasons we feel that locally created presentations offer advantages impossible with films.

Conclusions

Computer-generated graphic displays are of great value in teaching a wide variety of mathematical concepts which arise in the undergraduate curriculum. They are especially valuable for those concepts which require a degree of precision to illustrate or which have a dynamic aspect.

Not all topics benefit from the use of computer-generated graphics. Much of the material taught in a course such as first year calculus is computational rather than conceptual. For such topics as computation of derivatives or antiderivatives graphic displays are of little value.

Graphic displays play the same role in mathematics lectures as experiments do in physics or chemistry lectures. Like scientific experiments they must be carefully prepared. The availability of graphic display equipment is important in helping the instructor design the presentation to fit a particular viewpoint or suit a given class. The in-class presentation, however, may be a videotape instead of on-line.

Our experience is limited to a few courses and perhaps to the most obvious examples. We are sure that there are many other examples for which computer graphics will be of value. We look forward to hearing of these from our colleagues.

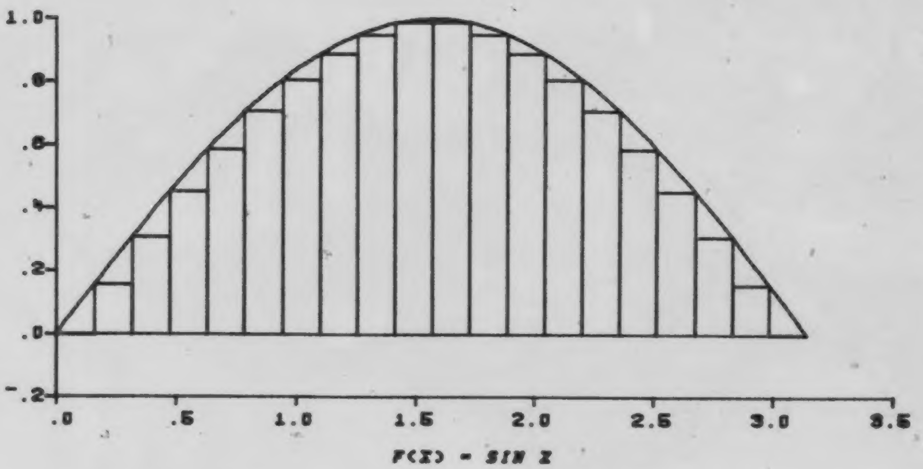
Examples

To illustrate the points made above, we have included hard copies of some applications. The Weierstrass function, the derivative picture, and the surface graph are all examples in which precision is required and there is no dynamic component. In each of these cases, an acetate transparency would be totally adequate as a teaching aid. The upper and lower sum approximations of the integral are examples where the precision really adds nothing to make the mathematics more comprehensible. A free-hand sketch would so as well.

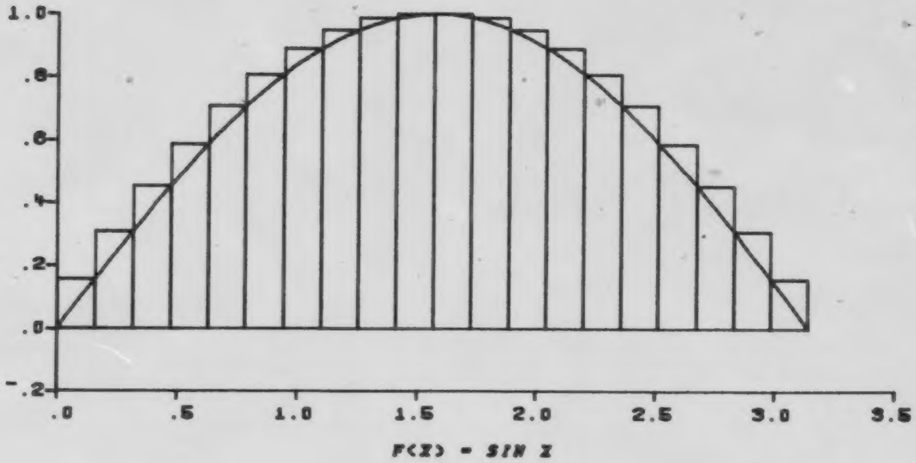
The Fourier series example is one where precision is required. Although no dynamism is inherent in the application, the picture becomes very complicated and difficult to comprehend unless you can see the next curve being drawn. An aspect of this is present in the harmonic motion example as well.

In the generalized Voltera-Lotka differential equation and in the Lissojous example, motion is an inherent part of the application. The static picture makes it impossible to tell in which direction the trajectories or parametric equation go.

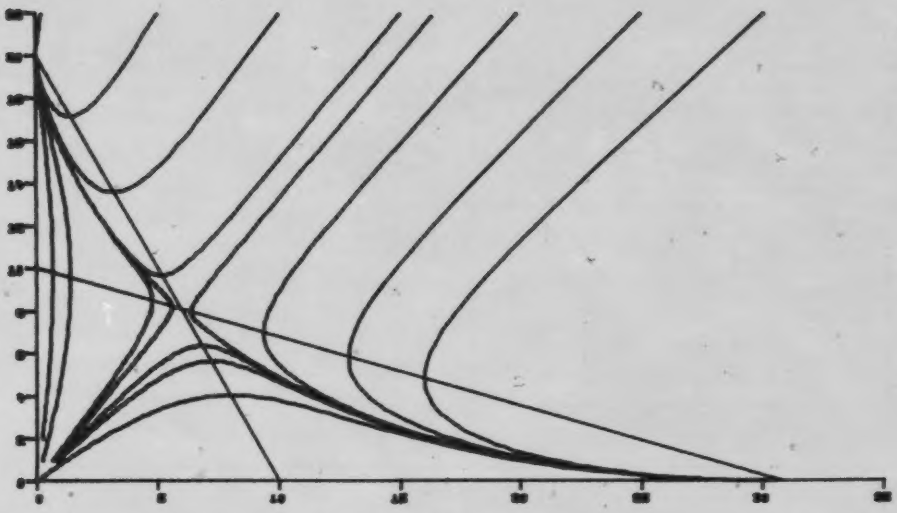
Finally, we have included two pictures of the surface, $z = y^2 + (x^2 - 1)^2$, one with hidden lines removed, one with them present. We think the removal of hidden lines makes a large difference in the effectiveness of the display.



THE LOWER APPROXIMATION IS 1.839



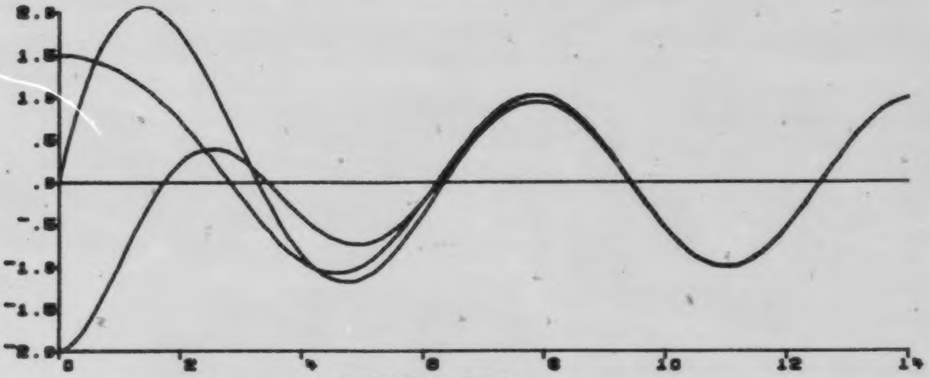
THE UPPER APPROXIMATION IS 2.153



SOLUTION CURVES FOR

X' = X * (0.2 - 0.0005 * X - 0.02 * Y)

Y' = Y * (0.2 - 0.01 * Y - 0.02 *X)

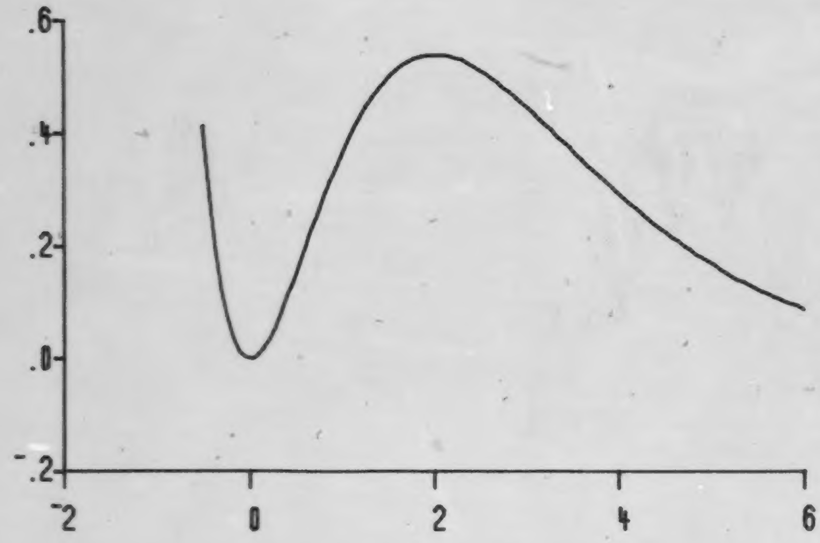


HARMONIC MOTION

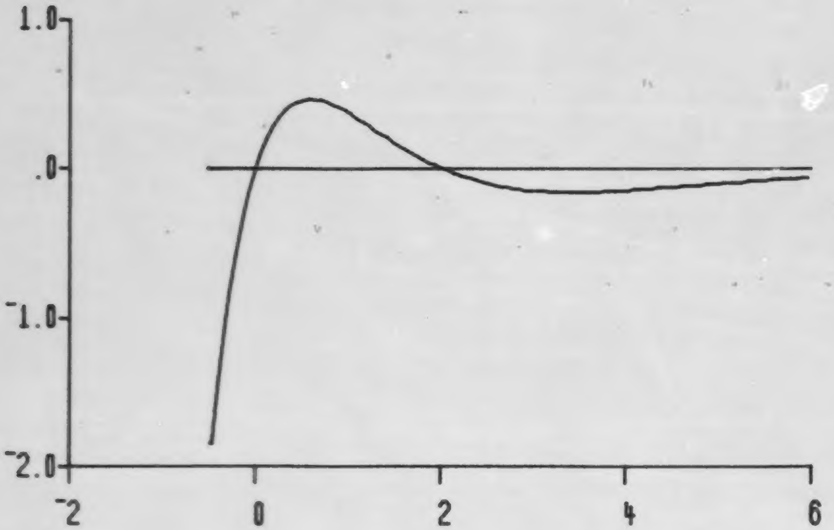
SOLUTIONS TO Y'' + A * Y' + B * Y = C * COS(X)

A = 1 B = 1 C = 1

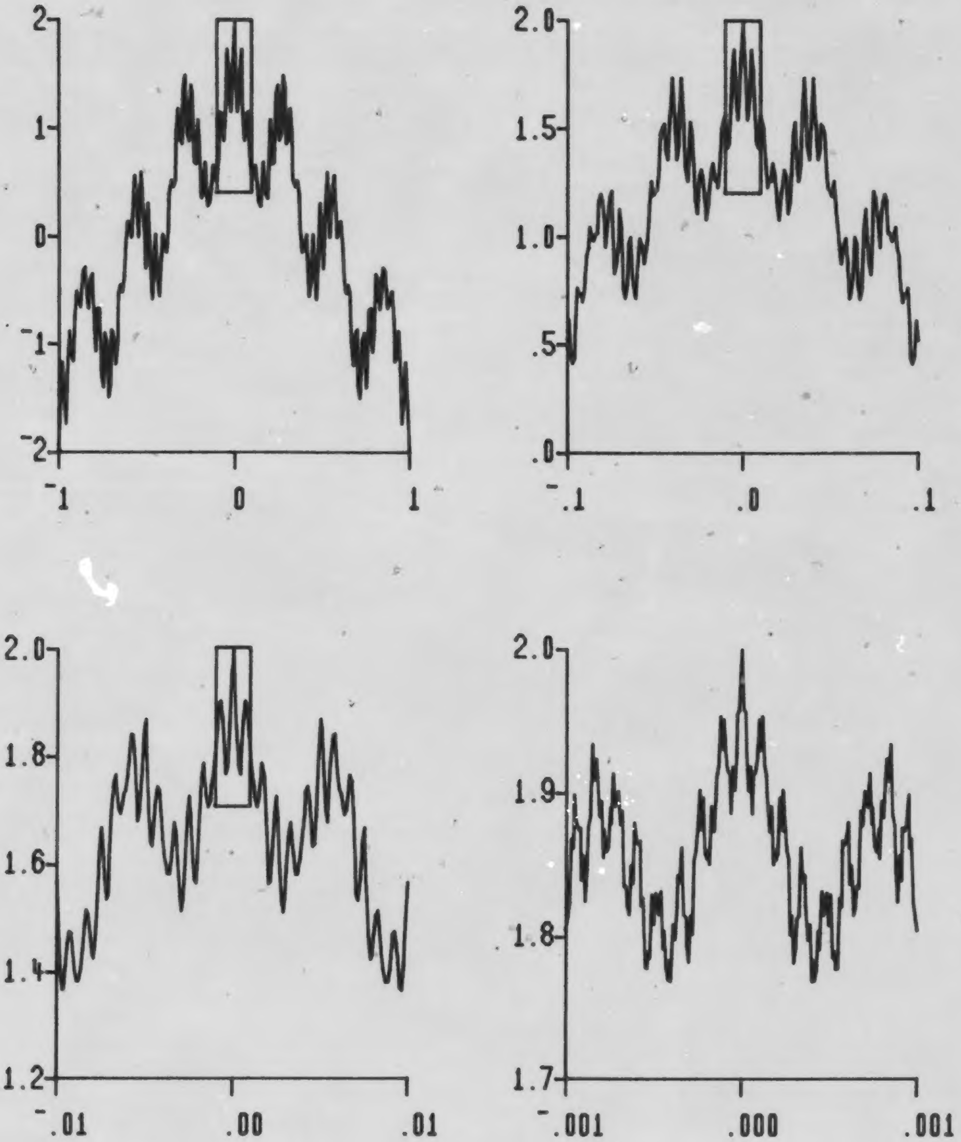
DIFFERENTIAL EQUATIONS



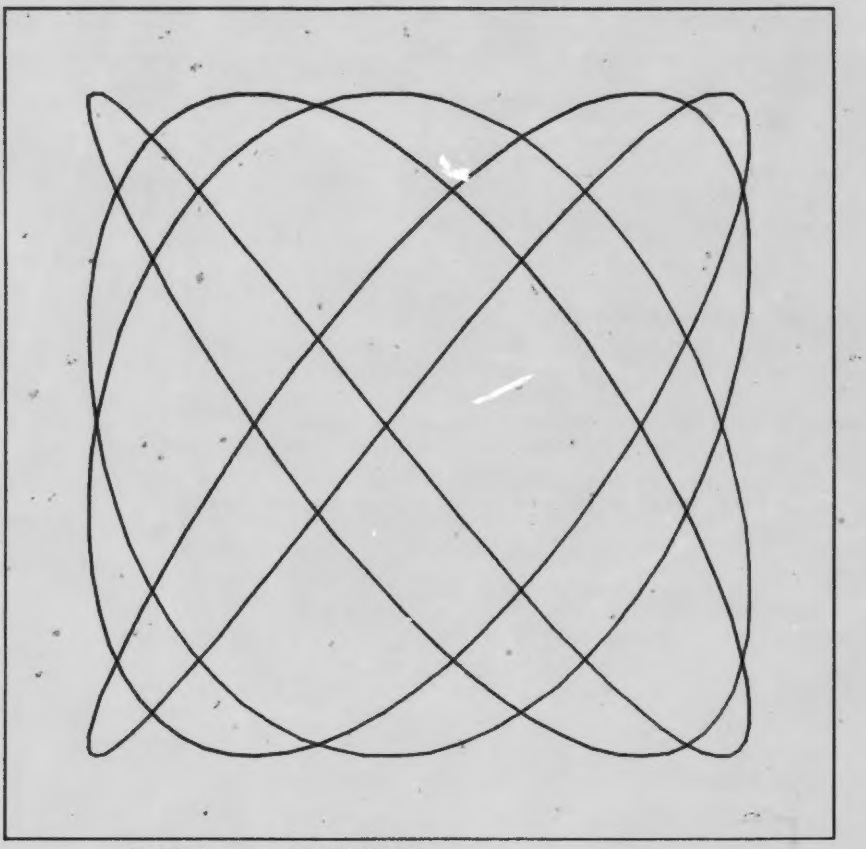
GRAPH OF F(X) = X×X×EXP (-X)



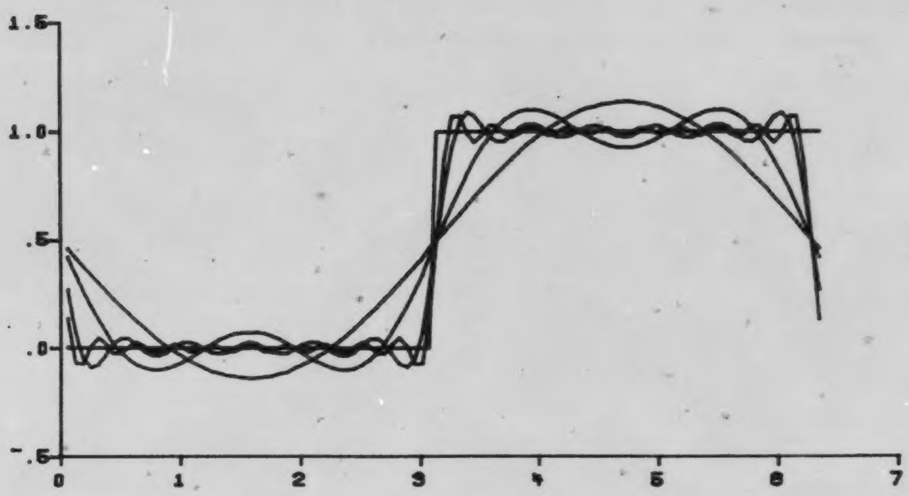
GRAPH OF F'(X)



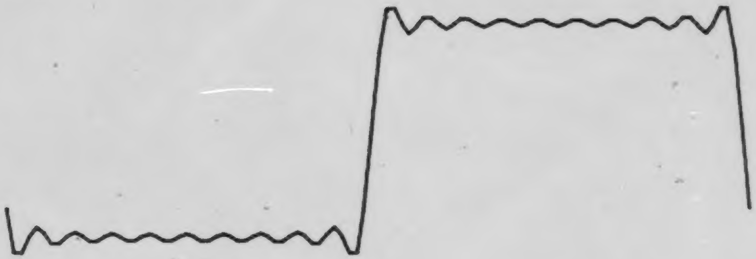
WEIERSTRASS'S EXAMPLE

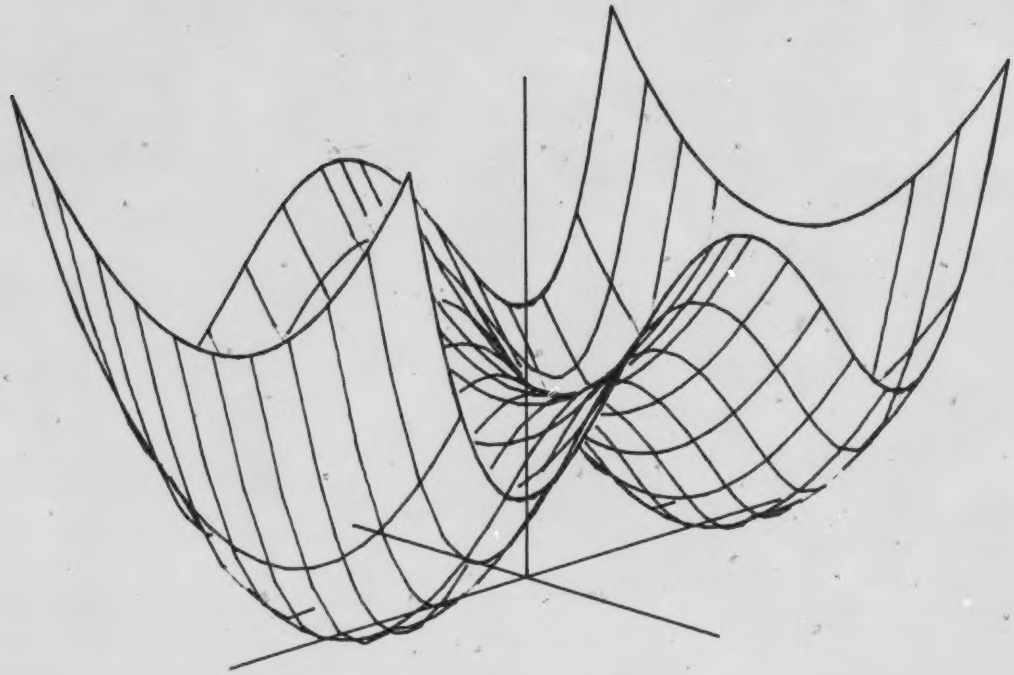


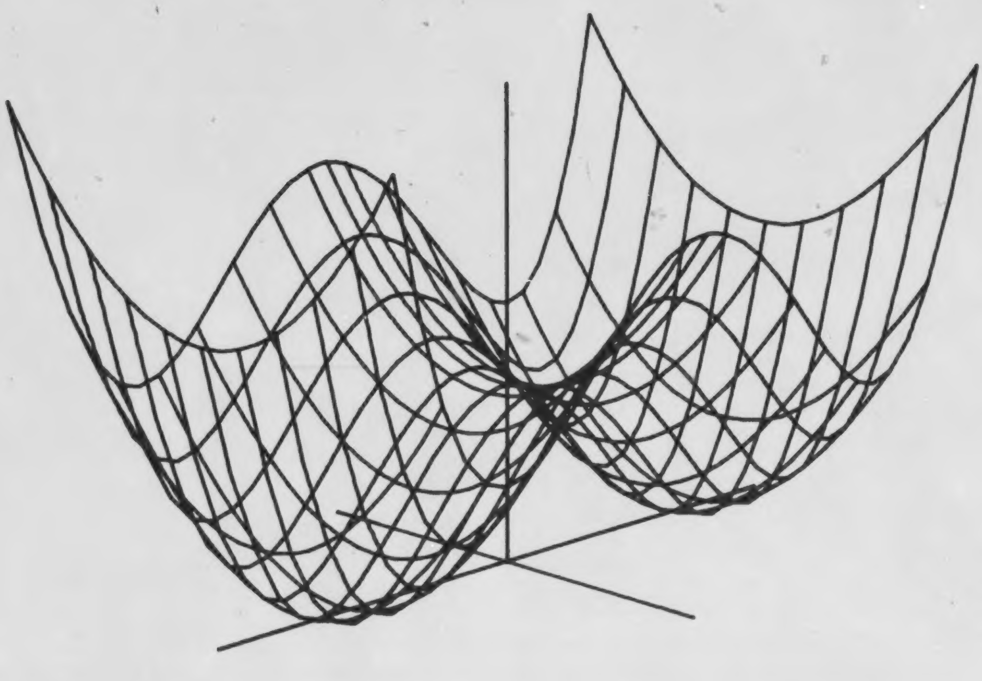
X(T) = SIN(T)
Y(T) = SIN(1.7+1.25×T)



PARTIAL SUMS OF THE FOURIER SERIES FOR Z**1







Language in the LOGO Computer Culture

Cynthia J. Solomon
Harvard University
Graduate School of Education
Cambridge, Ma. 02138

This paper discusses from an historical perspective some of the pedagogic considerations faced in designing the LOGO programming language. The programming language was to be one component of a new intellectual environment for students. Computers would also provide different programming contexts which students would explore through their own research projects. The language was to enhance the development of students' problem-solving skills. At the same time ideas, images, teaching metaphors and so on that could not be built into the language, yet were important to the process, were to become part of a metalanguage. Thus, within the intellectual environment encouraged by the computer's presence a common culture would be built up. In creating a computer culture three critical features were identified: the language used to talk to the computer, the content and context in which the programming was to take place, and the language used to talk about these activities (Papert and Solomon, 1972).

We intended to create a flexible medium in which students could interact with the system to pursue personal as well as public interests. The central focus of the culture was the enhancement of mathematical thinking in a broad sense by providing a place where general problem-solving skills could be developed through debugging programming projects (Papert and Solomon, 1970; Solomon, 1976; Solomon, 1978).

In 1966 a first version of LOGO was designed and implemented (Feurzeig, et al., 1969). In 1970 the first turtles were built. Turtles introduced many changes to the culture. It became infused with an attractive and powerful programming context and a simple set of primitive commands: the turtle could be told to go FORWARD or BACK a specific number of steps; it could be told to turn LEFT or RIGHT a specific number of degrees.

History

When we began designing the LOGO language in 1966 Seymour Papert, Dan Bobrow, Wally Feurzeig, Dick Grant, and I wanted to create a programming language for educational purposes with special emphasis on making the language easy for elementary school children to learn. We figured that if elementary school children could use the language, then older beginners could as well; but the ideas from such a computer culture would be maximally effective if encountered at an early age. Very nice, but what does that mean? What are educational purposes? BASIC, I am told, is easy to learn; it was becoming popular in 1966, why not use it?

At the time we were keenly aware of the problems of content and context and their influence on language. We had been observing children (elementary and high school) learning TELCOMP, an algebraic language similar to BASIC. Like BASIC the syntax was not complicated, but the context about which these languages centered was numerical. Kids who didn't like math to begin with or who had no introduction to algebra found the commands easy to learn but hard to apply.

The terminology used in most programming languages to talk about the process of programming was borrowed from different areas of mathematics. A good example is the word "variable." For beginners this word is confusing. For sophisticated users it is also confusing; its meaning seems to change in different contexts.

In the LOGO culture variable is replaced by a more powerful conceptualization, the idea of naming. Names can be given not only to things like words and numbers, but also to pieces of program, to procedures. Names are used to identify components of the problem and to clarify their roles. Naming thus becomes a thinking and debugging tool, and the flexibility of choice in kinds and lengths of names becomes part of the language design. In the tradition of other algebraic programming languages, TELCOMP (BASIC) assigned names in a way that did not demystify the process for beginners, e.g., LET X = 45. In early versions of LOGO names were given to things as in the following example:

```
CALL                   The programmer's typing
   THING: 45           is underlined.
   NAME:  "NUMBER.OF.THINGS"
```

In later versions we changed to

```
MAKE
   NAME:  "ANYTHING
   THING: 45
```

Another aspect of TELCOMP (BASIC) that we saw as another stumbling block in learning to program was the lack of procedurizing capabilities within the languages. They allowed some subroutinizing, but the names of these subroutines were numbers representing either a position in the program or in the computer's memory. The procedural characteristic of LOGO is not, of course, unique to it; other languages are as thoroughly procedural. In particular we were influenced by LISP, an extensible list-processor.

A procedure in LOGO has the following characteristics: (1) each procedure is a separate entity; it exists independently and at the same time as others; (2) each procedure has its own unique name which the programmer gives it; (3) each procedure can have its own private things or messages (local variables); (4) each procedure can be edited; (5) each procedure can be run separately from or by any other procedure; (6) each procedure can create and modify other procedures including itself; and (7) each procedure can be described recursively.

Although some of these characteristics are shared by more recent languages like PASCAL, the flavor of the language is different because of the restrictions placed on compiled languages like BASIC and PASCAL. In these compiled languages there is one program in the computer's active workspace rather than many procedures. The emphasis is on a hierarchical organization of knowledge rather than heterarchical; though the program may be subprocedurized in PASCAL (or PL/I), there must be a universal control structure operative all the time. LISP offers an alternative model for compilers. LISP procedures (functions) are compiled after they are debugged so that they can run more efficiently. The compiled functions then become just another part of the programming environment, not the domain itself.

We felt that the programming activities TELCOMP and BASIC lent themselves most easily to were calculator-like (at that time slide-rule like). We also felt that a more natural problem domain for children was one in which they played with words and sentences. Furthermore, our programming experiences in LISP had shown us the ease and elegance by which problems could be solved when divided into sub-parts; the individual procedures were then debugged separately and allowed to exist in the workspace. They could be ignored until they were invoked by the programmer or by another procedure.

We rejected LISP as a language for children and education because it had many formalisms that seemed to be unnecessary distractions for beginning students, and it had more features than we thought children would require. It also had a culture already attached that would have carried our work in a different direction, giving us less control and fewer chances to introduce fresh ideas. To us the overriding desire was to break with the past and to throw out terms which would inhibit beginners from getting into the process of programming and problem-solving. We wanted to design a different programming language in which certain ideas would take precedence, and we wanted to design a different meta-language by which we talked about the process of programming (Papert and Solomon, 1970).

LOGO: Some Bugs and Features

(The LOGO described here is the one currently available on a PDP-11 or LSI/11 from MIT's LOGO Group in the Artificial Intelligence Laboratory or from Societe Generale Tortue in Montreal. There are some differences between MIT 11-LOGO and BBN 10-LOGO which this paper does not describe. LOGO began at Bolt, Beranek and Newman. It was first implemented in LISP on an SDS-940. By 1968 it ran on a PDP-1 and then on a PDP-10. In 1970 I joined Papert at MIT, and the LOGO group at MIT was started. LOGO at BBN and LOGO at MIT followed different pedagogic influences, thus the language differences in the implementations can be traced to the differences in teaching philosophies, culture, etc.)

LOGO is a procedural, list-processing language. It is extensible to some degree. The kinds of things that are named are procedures, input messages, words, numbers, lists of words, turtle pictures, and so on. Names can be very long and have few restrictions except that they must be single words and not lists. There are two types of procedures: commands and operations. A command causes something to happen; an operation hands back a message to another operation or command. The distinction between a command and an operation can be illustrated by the difference in jobs between the turtle command FORWARD, which causes the turtle to move forward, and the operation SUM which adds numbers together and sends back the result to be used by another procedure.

Syntactically, procedures follow a prefix format with the exception of arithmetic operations which have two forms, e.g.,

PREFIX	INFIX
SUM 3 4	3 + 4
DIFFERENCE 3 4	3 - 4
PRODUCT 3 4	3*4
QUOTIENT 3 4	3/4
REMAINDER 3 4	3\4

With the introduction of infix notation parentheses were needed to control the interpretation of the statements. For example:

```
SUM PRODUCT 3 5 4 is unambiguously (3*5)+4 .
```

In earlier versions of LOGO (before MIT 11-LOGO) there were no decimals, only integers. Our original intention had been to treat numbers as words that had been given special meaning. PRODUCT was called MULTIPLY and QUOTIENT was called DIVIDE. In the first version of LOGO the arithmetic operations were limited to SUM, DIFFERENCE, and COUNT (length of words or lists). Other operations were derived from these either by the teacher or by the students. Turtles changed the emphasis; decimals were introduced. Now LOGO includes trigonometric operations in its initial vocabulary.

Naming Things

We wanted to distinguish names of procedures from other names and so we required non-procedure names to be prefixed by : (before DIVIDE / was used). For example, assigning names to things was at one time done by

```
MAKE
   NAME:  "MAGIC
   THING:  90
```

The programmer's typing is underlined. The computer's typing is not underlined.

"MAGIC is the name of 90.

A word prefixed by : (colon is called dots by LOGO users) for the thing that is named.

:MAGIC is 90.

```
PRINT "MAGIC
MAGIC

PRINT THING "MAGIC
90

PRINT :MAGIC
90
```

To intensify the distinction between names and things, we would play the dots-and-quotes game which draws upon the following joke: (Papert, 1969)

teacher:	say your age
child:	your age
teacher:	no, say your age
child:	10
teacher:	no, say " (quotes) your age
child:	your age
teacher:	great, now say : (dots) your age
child:	10
teacher:	right on

until the point of using quotes or dots to disambiguate meaning was clear.

In the process of implementing languages debugging takes place at different levels. Often, for example, the systems programmers add their own ideas about proper languages, the designers are inconsistent or are asking impossible things, and the original design gets changed. Compromises are made; often the result is positive, but in some cases it is not, as in the following. Somehow in a recent implementation of LOGO the long form of MAKE (as used above) was not implemented, instead the short form was made available.

```
MAKE "SUGAR "MAGIC
```

In this form the image conveyed by MAKE has not been satisfactory; Papert, for example, would like to return to CALL as the assignment command.

At the same time another form of assignment was added to the language, an infix form:

```
"SUGAR ← :MAGIC
```

Perhaps the real lesson to be learned here is that neither form is satisfactory, and the assigning of names to things must be reconceptualized until a form is found that relates to beginner's needs and systems programmer's needs and pedagog's aesthetics. One idea Papert and I have bounced around for a long time is having long forms (promptings or cuings) for all procedures so that their input messages could be asked for by the computer.

In a sense the implementation of ← , the infix assignment statement, represents the bastardization of language. It is a command and also an operation. It follows in the tradition of LISP in which every procedure outputs something. But it was added as a shortcut method to assign several names to the same thing, e.g., "X ← "Y ← "Z ← 0.

Another shortcut introduced to the language affected specific prefix procedures like PRINT, SUM, WORD, PRODUCT, SENTENCE, and LIST. In early versions of LOGO each prefix procedure took a specific number of inputs (including none) so that the parsing algorithm in the system didn't have to be cued by left- and right-sided markers (like LISP's parentheses). A change was introduced so that certain procedures could be given an indefinite number of input messages provided the statement is enclosed in parentheses.

```
PRINT (SUM 5 6 7)        PRINT (WORD 5 6 7)
18                       567
```

In the case of ← we see a need for rethinking how to set up an initial state of a program or micro-world. Another need in the language is a method to allow users to make procedures that take an indefinite number of input messages. A further extension would allow users to state different default conditions.

Naming Procedures

In the design of LOGO, both the language and the culture, there was an attempt to encourage humor and play as ingredients in problem-solving. One manifestation of humor and play was in the way we chose to name procedures. Commonly programming languages are looked at as formal instruments to rigorous, logical thinking, and attached to them is the notion that people are rigorous, formal, logical beings or that these qualities are necessary prior to participating in problem-solving situations. The implication is that fuzzy, intuitive thinking is always in juxtaposition to analytic; in fact most of us use both kinds of thinking depending on our familiarity with problem-solving and the particular context. This attitude is made all too real in most computer systems and programming languages where the process of debugging must take place outside of the programming language through other methodologies and structures.

The LOGO culture tries to capitalize on humor and play and to encourage both styles of thinking. We chose as our image of procedural description the bake-the-cake metaphor. We wanted to emphasize that procedures were descriptions. We thus rejected the term DEFINE as a way of beginning the description because of its link to mathematical formalisms. Instead we chose the preposition TO, in the sense of how to do something and a sign of dedication in the naming sense: TO FIND something you have to start looking, we dedicate the following TO JOE, our friend, etc.

A recent article on the problems of translating LOGO into Portuguese (Curado and Valente, 1978) interprets our use of TO as a way of turning nouns into verbs. In a sense this would be the computer scientist's way of looking at language. Given this interpretation, the authors quite legitimately point out inconsistencies within LOGO whether certain words can be thought of as verbs and whether it is perhaps the linguistic nature of the English language and American culture which allows this. Our original intention had been to convey the power of language to children in the sense that they, the children, the programmers, could give words meanings. Theimportant element was not whether a procedure was a verb or a noun, but rather whether it was a command or operation. A consideration in choosing names for procedures was whether they described the procedure's job.

The same article points out that in Edinburgh-LOGO, DEF is used to define procedures, not TO. Perhaps this illustrates a difference in American and British humor, and our deliberate move away from the formality of most programming languages is not very relevant to them. What is important is what this illustrates: that debugging teaching images and metaphors needs to take place in different contexts.

We wanted to emphasize the processes or reasons for the procedures. We wanted to encourage children to use names as problem-aiding tools. We have names or identify objects by names so we can talk about them and distinguish them from one another, and thus we can establish a common base of understanding. So in teaching children to program, we wanted to encourage naming things and not restrict unnecessarily the form of the word used as a name. Names could be long and silly like

```
8TEEN    FOOP    JOHN.AND.MARY.AND.JACK
```

We, of course, encouraged children to use names that were descriptive, but more important, we wanted children to give words meanings. One way of doing that was by creating procedures. Procedures were recipes or descriptions of how to do something or how to make something. We distinguish between procedures that cause something to happen and procedures that compute some piece of information and pass it back to another procedure. Thus, a command can stand alone while an operation cannot. Here are examples:

```
OPERATION              COMMAND

TO POW                 TO WOW
1 OUTPUT SUM 51 39     1 PRINT SUM 51 39
END                    END
```

Just saying POW to the computer would result in the computer saying

```
YOU DIDN'T TELL ME WHAT TO DO WITH "90
```

We could have said

```
PRINT POW
```

and then the computer would respond

```
90
```

Of course, WOW does this. Saying WOW gets a response of 90 from the computer, but that response cannot be used by another procedure.

These two examples do not bring out a problem which does arise in LOGO in clarifying these two types of procedures. The problem is due to the difference in syntactic rules governing these types of procedures. In more complex procedures this syntactic disparity becomes a conceptual problem. The command OUTPUT performs two jobs. It sends back a message and it stops the process. The only other command that stops the procedure naturally under program control is STOP. STOP is a command which does one thing, but it is never used in operations, only in commands. In any new implementations of LOGO the consequence of restricting OUTPUT to sending out messages and STOP to stopping processes should be examined.

Elaborating the Language

The bake-the-cake metaphor for defining procedures seemed to work with children in fifth, sixth, seventh grades and beyond. The image broke down with first graders. In 1973 I started to work with first graders. Several bugs in the LOGO culture became magnified; this one is probably the most important example. (As already mentioned it arose again when LOGO was translated into Portugues.) When working with students I had always emphasized that the activity we were engaged in was teaching the computer a new word, adding to the LOGO vocabulary, expanding what the computer knows. Sometimes new meaning is given to words which in other contexts take on quite different meanings; other times new words are created and then given meaning. By defining procedures we were teaching computers how to do things. TO was the definitional command. With first graders I expanded on this idea of teaching and made the teaching process more explicit by providing them with a TEACH procedure written in LOGO, but invoked by them whenever they wanted to create a procedure of their own.

The LOGO syntax for procedure description is step-oriented. Each step must be numbered; the number signifies the sequence in which the steps are to be interpreted. For example, to create a procedure called HI we say

```
TO HI
1 PRINT "HELLO
2 PRINT "THERE
END
```

and then every time we say HI the computer responds

```
HELLO
THERE
```

and not

```
THERE
HELLO
```

The metaphor of recipe makes the use of TO concrete for most older kids. For others, especially very young ones, it is not sufficient. Another confusion which became particularly apparent by working with very young children was the multiple role of numbers. Numbers were used to control the sequencing of instructions in procedures; numbers were also used as inputs to turtle commands as entities to manipulate, as in arithmetic. The TEACH procedure recognized these difficulties and was a contact point between the demands of the LOGO world and the children's world. Here is an example of how it worked. (Remember it too was written in LOGO.)

```
TEACH
TEACH ME HOW TO FUN
STEP 10:  FD 50
STEP 20:  RT 90
   .
   .
   .
STEP 90:  END
NOW I KNOW HOW TO FUN
```

The programmer's typing is underlined.

TEACH assigned steps numbers to the children's procedures. This use of number is perhaps an outdated tool in a graphics setting. In any new implementation of LOGO visual text pointers and other graphics aids for editing and writing procedures should be encorporated.

Although TEACH has not become a permanent part of the LOGO language the teach the computer metaphor has become a part of the meta-language. Looking at problems in terms of how one can teach the computer or another person helps to rethink the problem and shift it from something which one person has to do alone to a task which can be shared and broken into sub-parts. The sense is conveyed that problem-solving in a computer culture is a scientific investigation in which the process of arriving at solutions is as important as the solution. That the acquisition of knowledge is procedural and consists of teaching, learning, and debugging sub-pieces, then fitting them together and further debugging them so that they work together, is also part of the culture. Subprocedurizing and debugging apply to people as well as computer cultures.

TEACH exemplifies an idea embedded in the LOGO culture and expressed in the language. Users should be able to construct programming aids in the LOGO language. They should be able to add to a growing repertoire of building blocks and these procedures should not impinge on the user's world anymore than LOGO's initial vocabulary does. Furthermore, each user should be able to have a set of building blocks which are quite different from any other person's. The ability to create different user worlds within the LOGO language is one way to realize the computer's potential as a powerful and flexible aid to human problem-solving.

References

Curado, F. and J. A. Valente, "Some Problems of Translating LOGO to a Romance Language," ACM SIGCUE Bulletin, 12, no. 4, October 1978.

Feurzeig, W., S. Papert, M. Bloom, R. Grant, C. Solomon, "Programming-Languages as a Conceptual Framework for Teaching Mathematics," Report No. 1889, Bolt, Beranek and Newman, Inc., Cambridge, Ma., Nov. 30 1969.

Papert, Seymour, "Curriculum Units," 1969 (unpublished).

Papert, Seymour and Cynthia Solomon, "NIM: A Game-Playing Program," LOGO Memo #5, MIT Artificial Intelligence Laboratory, Cambridge, Ma. 02139, Jan. 1970.

_____, "Twenty Things to Do with a Computer," Educational Technology, XII, 4, April 1972.

Solomon, Cynthia, Problem-Solving in an Anthropomorphic Computer Culture, A.M. Thesis, Boston University, May 1976.

_____, "Teaching Young Children to Program in a LOGO Turtle Computer Culture," ACM SIGCUE Bulletin, 12, no. 3, July 1978.

A Comparison of the Problem Solving Styles of Two Students Learning LOGO: a Computer Language for Children[1]

Daniel H. Watt

MIT LOGO Group
Brookline Public Schools

I. INTRODUCTION

This paper is one of a series of reports resulting from an 18-month study of an elementary school computer laboratory.[2] The study, funded by the National Science Foundation, and carried out by the MIT LOGO Group in collaboration with the Public Schools of Brookline, Massachusetts, provided fifty sixth-grade students with betwen 24 and 36 hours of computer time and instruction in LOGO, a computer language designed for use by elementary school children. The work of sixteen of these students, representing a range of interests, work styles, and abilities was monitored in great detail. From the data collected we have prepared a series of reports outlining in detail the work of each child, and analyzing the learning of all sixteen children in the areas of computer programming, mathematics and problem solving. [2,3]

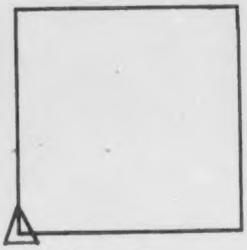
The contrasting work of Donald and Deborah, two of the sixteen students is presented in this paper. The contrasts between the students give rise to a number of pedagogical questions and issues which will be considered in the concluding section.

The computer language LOGO was developed approximately eleven years ago as a vehicle by which children could learn computer programming, problem solving and mathematics in a learning environment; it grew out of the cognitive psychology of Jean Piaget and an approach to the design of computer languages based on ideas arising from work in Artificial Intelligence.[5,6,7] Children are typically introduced to LOGO by using the computer to control a TURTLE, an imaginary creature which lives on a graphics display screen, the movement of which is controlled by commands typed at a keyboard; for example: FORWARD 100, BACK 50, RIGHT 90, LEFT 45, etc.

FORWARD 100 moves the TURTLE forward 100 TURTLE steps and leaves a trace on the display screen; LEFT 45 causes the TURTLE to rotate 45 degrees to the left. Drawing pictures with the TURTLE is an initial programming activity with an immense resonance for most children. Children can identify with the TURTLE, imagining themselves going through its motions as it carries out a particular task. At the same time, controlling the TURTLE becomes a metaphor for controlling the computer itself -- like the TURTLE, the computer responds to an ordered series of commands and to procedures, defined as series of commands.

The way in which the action of the TURTLE can lead to a geometric design, as well as the method used by children to define procedures is illustrated in the following simple examples. Having drawn a square by repeating the steps, FORWARD 100, RIGHT 90 four times, a student can define a procedure by choosing a procedure title (BOX, for example) and typing a series of commands in order.

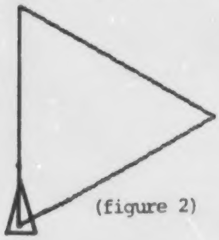
```
TO BOX
10 FORWARD 100
20 RIGHT 90
30 FORWARD 100
40 RIGHT 90
50 FORWARD 100
60 RIGHT 90
70 FORWARD 100
80 RIGHT 90
END
```



(figure 1)

When the student types the new command BOX, the shape shown in figure 1 will be drawn on the display screen. (The small triangle shown in figure 1 represents the TURTLE.) A similar procedure, TRI, could be defined as follows:

```
TO TRI
10 FORWARD 100
20 RIGHT 120
30 FORWARD 100
40 RIGHT 120
50 FORWARD 100
60 RIGHT 120
END
```



(figure 2)

Having defined procedures BOX and TRI, the student can now use these as commands to the computer, in effect developing his own private language as he goes along. In particular, they can easily be used as subprocedures in creating the drawing HOUSE, or the abstract design, FLOWER.

```
TO HOUSE
10 BOX
20 FORWARD 100
30 RIGHT 30
40 TRI
END
```

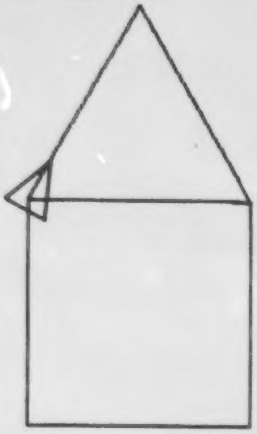
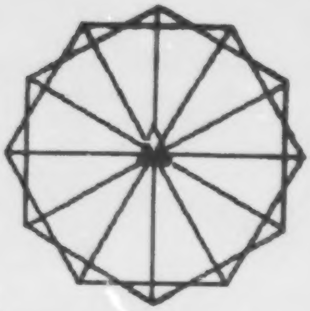


Figure 3



```
TO FLOWER
10 REPEAT [TRI RIGHT 30] 12
END
```

Figure 4

These TURTLE commands, when embedded in a suitable interactive, procedural computer language such as LOGO, provide the basis for a rich universe of activities called Turtle Geometry, which includes cartoon drawings, simple and complex geometric designs, mathematical theory building, and computer games and animation. Extensions of Turtle Geometry have proven fruitful when used with advanced high school students or MIT freshmen and sophomores.[8] The universe of Turtle Geometry provides a conceptual framework for such aspects of mathematics as coordinate systems, positive and negative numbers, the use of variables, symmetry and similarity, the significance of special angles (30-60-90, 180, 360, etc.). The computer programming involved in beginning LOGO activities can include the notion of procedures and subprocedures, recursion and iteration, the naming of procedures and variables, the use of conditional logic in the form of simple stop rules for repetitive procedures, the understanding of procedural hierarchy, and the use of debugging strategies.

Within the universe of Turtle Geometry there is room for different students, working individually, to create their own sub-universes or microworlds with their own limited but expandable set of concepts, and related activities and projects. The pedagogy of a LOGO classroom can be seen as the task of helping each child to create, explore and extend his or her own microworlds. In this paper we describe the microworlds of two students, Deborah and Donald, who approached a very similar project in very dissimilar ways. We feel that by focussing on these two individuals we can demonstrate how the computational environment of LOGO fosters the learning of children with very different developmental levels, learning styles, and academic abilities.

II. A Comparison of Donald and Deborah's Work

Deborah and Donald each carried out one major project as part of their LOGO experience. They each programmed the computer to draw a cartoon-like head. Both projects and the superprocedures used to draw them are shown in figure 5.

```
TO RABBIT
5 HAT
10 LITTLEEYES
15 FACE
20 PENUP
25 FORWARD 70
30 FORWARD 3
35 RIGHT 20
40 PENDOWN
50 RIGHT 90
55 RIGHT 90
60 FORWARD 3
65 FORWARD 60
70 FORWARD 5
75 FORWARD 3
80 RIGHT 90
85 RIGHT 20
90 EARS
END
```

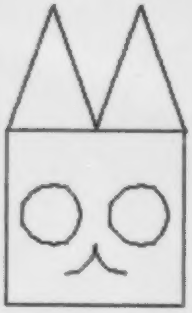


Figure 5a

```
TO HEAD
1 BOX
2 EYES
3 NOSE
4 MOUTH
5 BEARD
6 HAIR
70 EARS
80 HAT
85 FLOWER
END
```

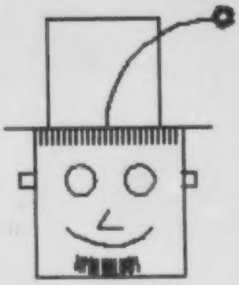


Figure 5b

Deborah is considered by her teachers to be a slow learner. In her regular classes she often appears to be withdrawn, indifferent to the subject matter or to her fellow students. When she began work in LOGO, she was totally dependent on the teacher -- requiring his reassurance on matters as routine as when to type a carriage return.

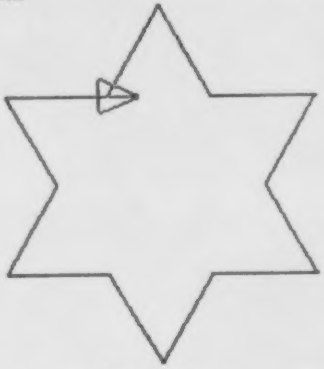
Deborah was able to slowly build her confidence and understanding by limiting her choices of LOGO commands and inputs, limiting the goals of her work, and by working in a way that minimized the chances of error. While Deborah began her LOGO experience by asking for help at every turn, she had a deeply ingrained resistance to new ideas or concepts. For a long time she rejected the use of subprocedures, which could have greatly expanded her possibilities. It was as though she deliberately provided herself with a very definite and restricted microworld in which to operate.

Deborah used as few different commands as possible in her work. Basic TURTLE commands along with arc primitives RARC and LARC were almost the only commands she used. For inputs to TURTLE commands, she used only multiples of 10, up to 100. If a larger effect was needed, she would use two steps, as in FORWARD 90, FORWARD 30.

Deborah began by using only inputs of 30 for all TURTLE commands, and gradually expanded to include other numbers, although 30, 60 and 90 continued to be her favorites. This microworld which Deborah chose for herself is very nearly as rich as all of Turtle Geometry. It includes squares, triangles, circles, stars, men, rabbits, and a variety of abstract designs, as well as the mathematical concepts of perpendicularity, inverse operations, the total turtle trip theorem, symmetry, similarity, estimation of lengths and angles, planning and debugging, and procedure writing.

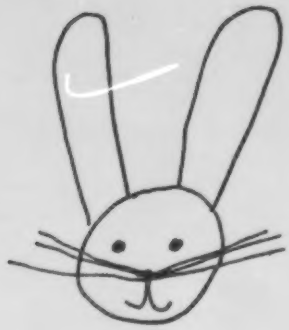
By limiting her inputs to numbers such as 30, 60 and 90, Deborah actually enhanced the possibility that random explorations would produce interesting results. At the same time, she seemed to have a high degree of visual intuition, often choosing precisely the correct input to produce a desired effect. Her patience in a one-step-at-a-time mode of operation was remarkable. Her format was quite stereotypic. (1) carry out one turtle step (turn, move or penup); (2) check to see if that looks right on the screen; (3) if so, write down the step and continue; (4) if not, clear the screen, retype all the steps previously written down and try another choice for the qustionable step; (5) when a design looks good, choose a name for it, and copy all the steps to make a procedure.

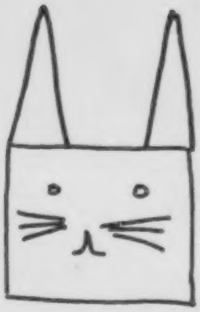
As an example of Deborah's success with this approach, we examine the way in which she drew a six-pointed star without making a single mistake. She began by turning the turtle RIGHT 30, and used a combination of FORWARD 70s and RIGHT 60s to complete the star. The actual rotations required to construct the star were RIGHT 120 at each point, and LEFT 60 at each inner vertex. The way Deborah accomplished these rotations was quite typical of her work. After each forward step, Deborah would turn the TURTLE RIGHT 60. She kept turning it RIGHT 60, until the TURTLE was headed in what seemed to her to be the right direction. This required two repeats of RIGHT 60 at each point, and five repeats of RIGHT 60 at each inner vertex. At one inner vertex she missed the correct orientation, and calmly repeated RIGHT 60 a total of eleven times until the TURTLE was aimed in the right direction. When she copied the steps in her notebook, she copied all eleven RIGHT 60s without any hesitation.



(figure 6)

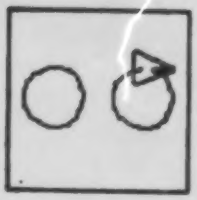
It was not until quite late in the series of classes that Deborah was ready to undertake a major project. She drew a picture of a rabbit in her notebook, and asked the teacher if he thought that would make a good project. He suggested modifying the rabbit, to make use of straight rather than curved lines, and helped her redraw the picture in a more simplified form.





(figure 7)

Although Deborah began by trying to draw the rabbit as a long series of commands, she soon accepted the suggestion that she break the problem into parts, and make each part a separate subprocedure. Although her work was directed toward an overall goal and involved a certain amount of top-down planning, she constructed each part of the rabbit piece by piece in an exploratory fashion. Once again her choice of inputs to FORWARD and RIGHT commands were such that it was relatively easy for her to make the design come out the way she wanted. Without any apparent planning, she chose the length for the sides of the rabbit's head (FORWARD 90 FORWARD 30) in a way that made it easy for her to locate the eyes and nose symmetrically. To locate the eyes she positioned the TURTLE at one side of the head, and moved it FORWARD 30. Deciding that that looked like the right spot for an eye, she drew a circle of radius 20. She then moved the TURTLE twice more in steps of FORWARD 30, decided that that looked about right for the second eye. Thus she was able to locate both eyes symmetrically. It was apparent that she had not decided that the rabbit's head was 120 steps across, and that she could therefore locate the eyes symmetrically 30 steps in from each edge. More simply, she knew that 90 and 30 were useful numbers and that if she kept trying FORWARD 30 it ought to come out right.



(figure 8)

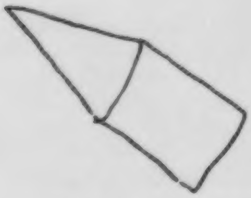
With the completion of her rabbit project, Deborah had almost totally reversed her initial feelings of dependence and incompetence. She invited her parents, teachers, and school principal to visit the computer lab, and in many ways, demonstrated to her visitors and classmates her new found sense of confidence, satisfaction, and power.

Donald's work, which provides a striking contrast to Deborah's, was characterized by a strong component of advance planning, and the creating of structures within which problems could be solved. At the same time, Donald was quite ineffective at the visually-based, exploratory modes of problem solving which were so useful to Deborah. He had difficulty estimating angles, and making use of the visual feedback provided by his explorations to improve his next attempt.

Throughout his work Donald was extremely receptive to suggestions from the teacher, often making use of new ideas before he fully understood them. In this way, he was able to incorporate strategies that would continue to prove useful as he gradually came to understand them through use in more than one context. He seemed to have the confidence that he could make use of the teacher's suggestions effectively and that he would eventually understand them, even if the concepts were a bit hazy at first.

As an example of both the effectiveness and the difficulties associated with Donald's structured planning approach as well as the frustration he experienced with visual approaches to problem solving, we consider his construction of a house from a square and a triangle, a common LOGO task, tackled by many students at an early stage of their LOGO experience.

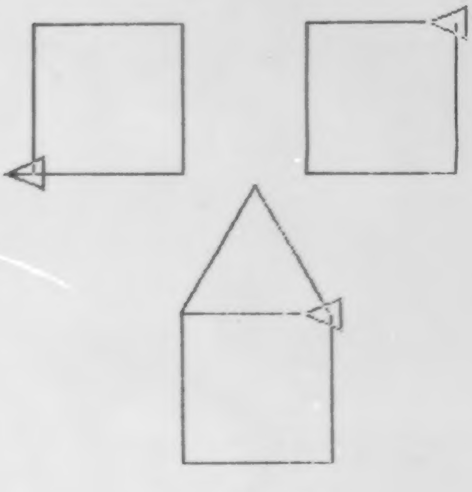
At first Donald attempted an exploratory approach to solving this problem. He began by having the TURTLE draw a triangle on the screen, making use of the TRI procedure described above. Having started by drawing a triangle, Donald established a framework for solving the problem corresponding to the initial orientation of the triangle. Asked to draw a picture of what he was trying to accomplish, he made the diagram shown in figure 9.



(figure 9)

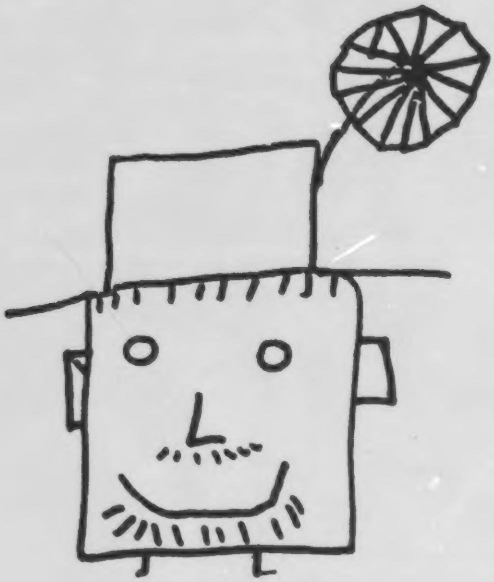
Since he was now dealing with two disorientations, the gap between the TRI and BOX procedures, and the tilted orientation of the entire shape, Donald had more difficulty than he could handle, and in an hour of exploration, never succeeded in resolving this problem.

At the next class, the teacher suggested that Donald draw the BOX first. This suggestion provided him with enough new insight to devise a plan for solving the problem. Donald avoided the usual problem of finding the rotation needed to attach the triangle to the upper left hand corner of the box. He moved the turtle to the upper right hand corner of the box, reversed its direction, and then drew the triangle so that its first side was along the top of the box. In this way he resolved the problem analytically, eliminating the need for visual explorations.



(figure 10)

Donald's HEAD project occupied him for more than 12 class periods -- more time than any other student devoted to a single project. This project also began with a plan, a cartoon-like drawing of a man's head. After one session of exploratory work, Donald drew a revised plan, and worked with his teacher to create a superprocedure designed to draw the entire figure.



(figure 11)

Donald's original superprocedure, TO HEAD, included the first six features of the head: the outside (BOX), EYES, NOSE, MOUTH, BEARD and HAIR. Once the superprocedure was written, each subprocedure became a mini-project, requiring one or two classes to resolve. Each feature of the head required it's own construction plan, a combination of the analysis and exploration needed to carry it out. While working on the features of his head, Donald made use of a great deal of teacher asistance -- especially in developing approaches to geometric analysis that were necessary to overcome his difficulties with visual problem solving.

When Donald came to his subprocedure, EYES, for example, he could not resolve the problem of how to locate the eyes symmetrically, by means of visual exploration. After several unsuccessful attempts, he asked for advice from the teacher. The teacher helped him set up a kind of grid with which he calculated the location of the eyes, and he was then able to solve this problem without any further difficulty.

Other, more complex problems that Donald encountered in the course of the project were resolved in similar fashion -- by creating structures which enabled him to solve the problem analytically and avoid visual explorations. In the course of his work Donald encountered estimation of distances and angles, the geometry of arcs and circles, the total turtle trip theorem, and the use of both grid-based and intrinsic coordinate systems. He learned to use subprocedures and sub-subprocedures, to use patterned procedures making use of a REPEAT command, to make use of variables to control the size and shape of his hat and flower and to use a procedure with a conditional stop rule. Although Donald only learned these approaches to the extent necessary to solve the particular problems inherent in his project, each succeeding use of the same concept reinforced his exposure to it, deepening his sense of mastery.

Donald's final figure, drawn by the superprocedure, HEAD, represents an almost literal mapping of his revised plan, figure 11, into a computer program.

III. Conclusion

In summary, we observe that Deborah's and Donald's problem solving s'yles were markedly different in several ways:

1. Planning
Donald was a planner. He made an overall plan for his project and created separate plans for each subpart of his project. Deborah had an overall idea of what she was trying to accomplish, but her actual project was carried out in an exploratory manner, one step at a time. Donald's superprocedure HEAD, was written to serve as a framework for his project; Deborah's superprocedure, RABBIT, was written as the final step in completing her project.

2. Analysis vs. Intuition
Deborah relied on an accurate visual intuition and a very limited set of inputs to TURTLE commands; Donald's visual intuition was poor -- he relied on mathematical analysis to resolve each problem he faced.

3. Comfort With New Ideas
Donald absorbed a great deal of new information and tried a number of new approaches as he carried out his project -- confident that he could learn the skills he needed as he went along; Deborah resisted new learning -- her learnings consisted largely of consolidating and reinforcing old skills and methods.

4. Obtaining Assistance From a Teacher
Donald made regular use of assistance from his teacher, while Deborah insisted on doing things in her own way, and resisted suggestions which she did not fully understand.

These observed differences give rise to pedagogical questions that are not limited to teaching children to use computers: How can teachers learn the learning styles of their students? How can teaching approaches in normal classes be flexible enough to support such a wide variation in learning styles? How can a teacher help Deborah acquire some of Donald's analytic approach? Can Deborah's intuitive understanding be learned by Donald?

The success of both Donald and Deborah in a LOGO environment leads us to believe that additional experience in such an environment may be productive in beginning to answer these questions.

FOOTNOTES

1. The work reported in this paper was supported in part by the National Science Foundation (nder grant number 77-19083SED) and conducted at the Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts. The views and conclusions contained in this paper are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the National Science Foundation or the United States Government.

2. "An Assessment and Documentation of a Children's Computer Laboratory, LOGO Memo 48, 1977.

3. "An Interim Report of the Brookline LOGO Project", LOGO Memo 49, 1978.

4. "Final Report of the Brookline LOGO Project, " to be published Spring, 1979.

5. "Programming Languages as a Conceptual Framework for Teaching Mathematics". Feurzeig W., Papert S.A., Bloom M., Grant R., and Solomon C. Report No. 1889. Bolt, Beranek, and Newman. Cambridge, Massachusetts. 1969.

6. "Twenty Things to do With a Computer". Seymour Papert and Cynthia Solomon. LOGO Memo 3, June 1971.

7. "Teaching Children to be Mathematicians vs. Teaching About Mathematics," Seymour Papert. LOGO Memo 4, July 1971.

8. Turtle Geometry, Abelson and diSessa, to be published by MIT Press in 1979.

Impact of Software Engineering on Computer Science Education

Chaired by Ken Magel
University of Missouri-Rolla

Software Methodology for Courseware Developers

Ken Magel, University of Missouri-Rolla

Abstract not available at time of printing.

MSE-79: A Master's Program in Software Engineering

Richard Fairley, Colorado State University

Abstract not available at time of printing.

Fundamentals of Computer Architecture and Computer Communications for Courseware Developers

Chaired by Dr. Retnadhas
Western Illinois University

A Course Structure: Computer Communications Subsystems

Kenneth J. Thurber, Sperry Univac, St. Paul, Minnesota

This paper describes a senior/introductory graduate student level first course on computer communications architecture from a distributed system architecture level. The course deals with switches, paths, and their system implications. Functional topology issues of distributed and network systems are discussed. The implementation of their data transfer mechanisms via paths and switches is considered from the viewpoint of their protocol and construction tradeoffs.

This course is based around seminar courses taught at the University of Minnesota in the Computer Science Department, a book to be published in June 1979 by D.C. Heath, and a tutorial developed for the IEEE Computer Society to be given at the first Annual International Distributed Processing Conference.

On Computer Alchemy

G. Jack Lipovski, The University of Texas, Department of Electrical Engineering

The art of hardware design is progressing through a revolutionary stage of development with the introduction of LSI in general and microcomputers in particular. An analogy is drawn to the progression which chemistry passed through with the introduction of large organic molecules. Just as quaint terms like oil of vitreol and urea had to give way to systematic names like 2,2 dihydrobenzol, so should hardware terms like Johnson counter and UART give way to systematic names and descriptions. Using these quaint names we are computer alchemists. But we are being forced to develop a more scientific approach to cope with those large organic molecules, the LSI modules. This articles develops the analogy and shows how hardware description languages should fit into the scheme of scientific design and analysis of digital systems. Several widely used languages are introduced, and an attempt to standardize them is discussed. From this perspective conclusions are drawn regarding the teaching of this science.

A Microoperation Approach for Teaching a Computer Architecture Course

Svetlana P. Kartashev, University of Nebraska-Lincoln

Steven I. Kartashev, Dynamic Computer Architecture, Inc.

C.V. Ramamoorthy, University of California-Berkeley

The diversity of types of architectures for computers and systems makes constructing an adequate computer architecture course problematic. The problem of course design will be aggravated in the future. If the current tendency to increase the number of different architectural types continues, the computer industry of the eighties will produce a large number of various supersystems with complexities equivalent to those for hundreds of conventional systems.

These supersystems will be equipped with adaptable architectures, capable of performing software-controlled adaptations toward executed algorithms. Thus a system with an adaptable architecture will have not a single, but hundreds of different architectures each of which will be distinguished by the number and sizes of concurrently operating computers. Each computer will be equipped with tens of different instruction sets, and the same hardware resource will be used for forming, via software, different types of architectures such as multicomputer, multiprocessor, array, pipeline, etc.

If now an instructor is confronted with difficulties in explaining basic concepts for computer architectures with an average throughput, then one can imagine the problems arising when teaching the supersystems with adaptable architectures. Therefore it is quite important to outline an approach for teaching a computer architecture course that makes it possible to study different architectures independent of their complexity and technological implementation. Designing such a course is a complex task and requires the collective efforts of many instructors and computer architects.

One possible approach is proposed that solves the problem since it introduces an axiomatic description of an architecture. This approach was named a microoperation approach because it is based on a single concept (axiom) called a microoperation.

The Avid System: A Multi-user Minicomputer-based System for Computer-Based Education

K.W. Scholz & D. E. Sloane
Advanced Interactive
Systems, Inc.

ABSTRACT:

A comprehensive CBE (Computer-Based Education) system which includes facilities for interactive graphics, natural language processing, and data file manipulation has been implemented on a DEC PDP-11 minicomputer. Lesson authoring aids permit inexperienced authors to create lessons with ease and permit transporting lessons to and from other systems. The system supports eight simultaneous independent users and provides facilities for student performance evaluation and record management.

1. INTRODUCTION

AVID (an acronym for Academic and Vocational Interactive Dialog) is a general purpose system for Computer-Based Education (CBE) implemented on the Digital Equipment Corporation (DEC) PDP-11 under the RT-11 operating system. Although AVID was developed in an academic environment, it has been designed for a wide range of instructional applications. Design of the system was motivated by three primary objectives.

A. To provide maximum responsiveness to the needs of the learner by including a comprehensive question-answering facility. The facility permits a student to enter a query as a simple English question in order to locate curricular material anywhere in the system's data base.

B. To permit lesson authoring by people with minimal computer experience. Lesson authoring is eased by patterning the authoring process on a typical dialog between two people. An AVID lesson is structured as a recipe for conducting such a dialog. The authoring facility which is based on this model permits a content expert with little or no computer experience to generate quality lessons with minimal training.

C. To be cost-effective and economically justifiable by designing AVID to run on a small minicomputer rather than time-sharing on a full-sized computer. AVID could service several terminals on a system as small as a DEC LSI-11 or a Heath H-11 for a total hardware cost under $5000.

2. AVID from the Student's Perspective

The mechanics of the student interaction with AVID have been kept as simple as possible. In addition to the traditional LOGON and LOGOFF functions found on any timesharing system, a student need only learn three simple directives which permit one to (1) exit immediately from the current lesson even if that lesson has not been completed; (2) back up to a previous display or "frame"; and (3) return to a previous lesson from which a student exited by asking AVID a question. Any line of input to the system is terminated by pressing the RETURN or ENTER key (depending on the terminal), and typing errors may be immediately corrected using a delete key.

Other than these few mechanical operations, interaction between AVID and the student consists of a simple English dialog. Immediately after LOGON or at any subsequent point in a lesson, the student may answer a question with another question, such as, "Do you have any lessons on Biochemistry?" or, "What information do you have on electronic theory? " AVID will attempt to locate information anywhere in its data base that appears to be germane to the student's question. If only a single lesson or portion of a lesson appears relevant, that information will be immediately displayed. A more general question which suggests a small number of specific topics (less than twenty) causes a numbered list of these topics be displayed for the student's choice. If a question is too general (e.g., "Can you tell me about medicine?") the student will be asked to be more specific.

Whenever a student question directs AVID into a lesson, the previous lesson or frame from which the question was asked is remembered in the event the student wishes to return to it. This memory facility is recursive to twelve levels. Once a student is satisfied that his question has been answered, a single-key command can be used to back up to each previous frame from which a question was asked.

The remaining student-system interaction is under the explicit control of the lesson author. Each frame in a lesson will present information as sentences, paragraphs, or graphics figures, and will solicit input from the student. The response may be a letter, a number, a phrase, a sentence, or a non-linguistic response mediated by cursor positioning. The lesson author will have provided a collection of response alternatives which AVID will use to direct the student to the frame appropriate to his response. This normal response-contingent dialog is maintained at all times unless the student chooses to deviate from it using the query facility described above.

3. The AVID Lesson Preparation Facility

The AVID Lesson Preparation Facility (LPF) provides two distinct methods of lesson preparation. In the first method, the LPF may be executed interactively by maintaining a dialog with the author much as a completed lesson communicates with a student. In the second method, the AVID

Text Editor (a facility which is totally independent of the LPF) may be used to prepare a lesson source file for subsequent compilation by the LPF. The source file may consist of an entire lesson, portions of several lessons, or corrections and additions to an existing lesson.

Lesson Structure

The conceptual view of a lesson is the same for either programming method. Each lesson consists of an indefinite number of frames, where a frame is the unit of information displayed at one time. One frame may comprise as little information as a screen erase or a single character, or as much as a screenful of text (24 lines of 80 characters) or graphics illustrations. From an author's viewpoint each frame must first be given a name. The frame's name or frame title constitutes an important component in AVID's data structures. Each frame title consists of a number of keywords which are ordered from most general to most specific. The most general or leftmost keyword identifies the lesson topic, and subsequent keywords refine the title down to the specific topic of the associated frame. In the event a frame represents a continuation of the theme of a preceding frame, integers (rather than words) are suffixed to the rightmost keyword. In any AVID curriculum, the frame titles organize the entire data base as a single semantic network.

Frame Structure

Each frame has three components, any of which may be omitted. Part one, the Text Block, consists of the text or graphic information to be displayed to the student. Part two, the Calc Block, consists of an indefinite number of lines of calculations, function calls, string manipulations, or conditional branches (within the Calc Block) entered in a syntax similar to BASIC or FORTRAN. Arithmetic expressions may be entered using fully parenthesized infix notation. The available functions include facilities for student record manipulation, interactive graphics, data file manipulation, natural language processing, and generation of random numbers and timed delays. Part three, the Response Block, contains a number of anticipated responses, each associated with the title of the frame to be displayed if that response is selected by the student. Various pattern matching options (disjunction, conjunction, wild cards, etc.) are provided to allow the anticipated response to match student spelling errors or synonyms.

The Authoring Process

If an author elects to use the interactive capability of the LPF, he or she will be explicitly prompted for each component of the frame. If the response to the prompt for a frame title is the name of an existing frame, the author has the options of ignoring copying, modifying, replacing, or inspecting that frame. If one elects to replace, modify, or create a frame, each of the three sections will be prompted. The author may respond to any prompt with a blank line, in which case that section will be terminated and the next section will be prompted. Any frame titles introduced in the Response Block will either be linked appropriately (if the frame already existed) or be added to a list of incomplete frames (if the frame does not exist). As frames are completed, the list of incomplete frames will be displayed as an aid to the author.

The commands permitted in a Calc block are very similar in form and function to existing algebraic programming languages. The AVID lesson author who is familiar with such languages (or who acquires such familiarity through use of AVID) can use this capability to produce interactive graphic displays, mediate interaction with complex data bases, provide extensive tracking of learner performance and progress, and expand the flexibility of response-contingent branching by allowing author-controlled manipulation and processing of learner responses. The effective use of this facility, however, requires that an author achieve a degree of programming competence that may be beyond his desire to acquire. The structure of any frame in AVID permits totally ignoring the Calc Block while still providing a versatile response contingent linkage to a number of other frames. Thus an inexperienced author can still generate sophisticated, highly interactive lessons by specifying only the Text and Response portions of each frame. Due to this simplicity AVID extends the capability for lesson authoring to a far wider audience.

4. AVID System Design Summary

AVID has been designed to run under Digital Equipment Corporation's RT-11 Operating System on a PDP-11 computer with at least 16K words of memory and one disk. The AVID resident monitor provides fully time-shared interactive lesson execution for up to eight students with a 28K word computer. A system with less than 16K of memory could provide either multiple student access or the Lesson Preparation Facility, but not both simultaneously.

Structure of the AVID Data Base

The data base is developed in two files on a mass storage device. The first, or Master File, consists of a header and a number of variable size records, one per frame. The second file serves as the Frame Title Library and is sequentially allocated with fixed size records. The Frame Title Library serves as a key or index to the Master File and provides indexed sequential access to the Master File for any search operations.

The AVID System Monitor

The AVID Monitor timeshares among all currently active ports by context switching among ports at each I/O wait. Port tables approximately 1.5K words long are associated with each port and contain essential address, flags, and I/O buffers. A context switch is effected by pointing a register to the appropriate port table then flowing into code common to all ports. Details of the context switching algorithm were described in Scholz (1977). The monitor design relegates all less frequently used functions to non-resident overlays. Whever the resident portion of the monitor identifies the need for one of these functions, the appropriate non-resident overlay is loaded into that student's port table. Once loaded, the overlay executes as a co-routine to the monitor and terminates by passing control either to another overlay or back into the monitor.

Natural Language Processing

Question answering is processed in two distinct phases. The first phase parses the student's sentence into an ordered collection of keywords which serve as a probe into the Frame Title Library. The second phase uses the probe to search the FTL in an effort to locate the desired information. In addition to its use in processing student queries, the language processor is available under author control through a series of Calc Block commands. The experienced author can use this feature to permit complex analysis of student responses.

The English parser uses a Markovian grammar which is represented as a path table (Hays, 1967). The table is based on an English vocabulary of around 100 common adjectives, articles, and verbs, and contains a list of legitimate transitions among them. The parsing algorithm is driven by the path table using a simplification of the transition network parser described by Woods (in Rustin, 1973).

The ordered keyword list or probe is processed by the second phase which performs the FTL search. The probe is compared keyword by keyword with each Frame Title and a match score, weighted by the length of the probe, is calculated. Those scores which exceed a criterion are retained as a list. When the FTL search is completed, the list is sorted by match score, and that frame whose title had the highest score is displayed as the answer to the student's query. If the top several match scores are nearly identical, the associated Frame Titles are listed to the student as a menu for his selection.

Despite the relative simplicity of the language processing algorithm, its performance as an information retrieval facility is extremely effective.

Student Record Management

Student access to AVID is controlled by a password file. This file contains each student's password, a set of permission bits, the time of day of the most recent logon, and the cumulative use time to date. In addition, a separate student record file is associated with each password. If desired, a lesson author may reserve file space for student record use by any lesson. AVID then reserves the same relative position in each individual student file for use by that lesson. The author can then include simple statements to save performance data in that lesson, and AVID appropriately maintains the information in the file of any student accessing the lesson. Utility routines can then produce listings of information on all lessons entered by a given student, or of all students who have entered a given lesson.

The AVID Lesson Preparation Facility

The Lesson Preparation Facility (LPF) is entered from any student port and can be used by only one port at a time. Its access is controlled by a queue in the AVID monitor which governs its sequential allocation to various authors on a first come-first serve basis.

Whenever the LPF is entered, it first scans the Frame Title Library in order to produce a listing of any frames that have been named but not completed. It then sequentially prompts the author for a Frame Title, a Text Block, a Calc Block, and a Response Block (see Section 3). As the information is entered by the author, space is allocated in the Master File by using an available storage pointer which is maintained in the first block. All expressions entered by the author in the Calc Block are converted from the parenthesized infix notation to a reverse Polish intermediate code using conventional compilation algorithms. The intermediate code is interpreted at execution by the AVID monitor. Listings for lesson debugging are produced by disassembling the intermediate code back to source format, thus eliminating the need to waste file space by storing the source code. The listing facility can re-create lesson source files which can be transported via magnetic tape or diskette to other AVID systems.

One of the AVID Monitor overlays is a Text Editor. This overlay is a line-oriented, contex editor designed to be used for lesson preparation. A source file created with the editor may consist of an entire lesson, a portion of a lesson, or corrections to an existing lesson. Source files created with the editor are submitted to the LPF for compilation. The LPF communicates the result of the compilation back to the author by creating a log file which indicates any errors detected during compilation.

The editor and the ability of the LPF to process source files greatly simplifies the process of lesson transportability. Although there have been several attempts to standardize lesson source listings in order to permit lesson interchange, there exist no universally agreed-upon formats (to our knowledge). The flexibility of the AVID editor makes it possible to modify CBE lesson source files obtained from other installations into a form acceptable to AVID. Similarly, an AVID source listing is easily read and understood, thus simplifying its adaptation to other systems.

5. Motivation for the Design

The pedagogical model on which AVID is based grew from the present authors' prior involvement with research on learning and memory. The investigation of semantic information structures (Quillian 1968; Rumelhart, Lindsay, and Norman, 1972) prompted our investigation of various approaches to the representation of information in memory (Scholz, 1972; Scholz and Potts, 1974; Potts and Scholz, 1975). This experience fostered our view of the educational process as a systematic incorporation of new material into previously established information structures. New information is not simply written into a passive memory store, but is actively associated or understood in the context of previously existing knowledge. While we recognize that understanding may be mediated by spatial, affective, or temporal as well as semantic associations, we feel that the single most important constituent in learning is the existence of a prior associative structure.

This view of learning places strong emphasis on the need to present information in such a way as to facilitate its incorporation into the student's prior knowledge of the universe. No single author or educator can possibly anticipate the optimal presentation sequence for every student. Rather, the student must be free to seek the information needed, motivated by current level of understanding. Thus, a student can best profit from computerized education if free to control the interaction rather than forced to conform to the author's ideas about the organization of information. The AVID system embodies this orientation both in the semantic organization of its data base and in the language processor through which it is interrogated.

Although this view of the learning process is neither totally profound or totally original, to our knowledge, AVID represents the first attempt to incorporate such a view into the design of a CBE system.

Summary

AVID, a general purpose CBE system implemented on the PDP-11, has been described from three perspectives. For the student, interaction with AVID can be directed either by student questions or by the sequential flow designed by the lesson author. For an author, a lesson can be created quickly and efficiently by using AVID's interactive lesson preparation facility. Alternatively, there is a library of specialized functions at the experienced author's disposal for use in creating graphics illustrations, maintaining student records, or performing arithmetic calculations. For the system designer, AVID's data base is developed in a sequentially allocated master file. This file is keyed by the Frame Title Library, a second file which associates each frame's title with its origin in the master file. AVID is a timesharing system which permits simultaneous independent interaction with up to eight students (on a 28K system).

REFERENCES

Hays, D. G. Introduction to Computational Linguistics. New York: American Elsevier, 1967.

Potts, G. R., and Scholz, K. W. "The Internal Representation of a Three-term Series Problem", Journal of Verbal Learning and Verbal Behavior, 1975, 14, 439-452.

Quillian, M. R. "Semantic Memory", in Minsky, M. L. (Ed.), Semantic Information Processing, Cambridge: MIT Press, 1968.

Rumelhart, D. E., Lindsay, P. H., and Norman, D. A. "A Process Model for Long-term Memory", in Tulving, E. and Donaldson, W. (Eds.), Organization of Memory. New York: Academic Press, 1972.

Scholz, K. W. "A Cognitive Approach to the Storage and Retrieval of Structured Verbal Material". Unpublished Doctoral Dissertation, Indiana University, 1972.

Scholz, K. W. "Timesharing on the PDP-11 under RT-11. Behavior Research Methods and Instrumentation, 1977, 9, 173-175.

Scholz, K. W. and Potts, G. R. "Cognitive Processing of Linear Orderings". Journal of Experimental Psychology, 1974, 102, 323-326.

Woods, W. A. "An Experimental Parsing System for Transition Network Grammars", in Rustin, R. (Ed.), Natural Language Processing, New York: Algorithmics Press, Inc., 1973.

THE ISU MICROSUPPORT NET: A PROTOTYPE SYSTEM FOR PERSONAL COMPUTING IN EDUCATION

Dr. Richard E. Horton
Associate Professor
Electrical Engineering
Iowa State University
Ames, Iowa 50011

INTRODUCTION

Decreasing costs associated with microprocessors and personal computers have added a new dimension of responsibility and activity to those concerned with computer systems in education. Teachers in the classroom or laboratory may already have access to their own personal computer. So may some of their students in their homes or study rooms. New opportunities for educational innovations with these rather sophisticated, yet low cost, computers will soon be limited by our imagination and creativity, not availability.

Feeling a need to bring order to the selection of available computers and accessories, the ISU Computation Center initiated a research project in 1977 to create a prototype educational computing system (the ISU Microsupport Net) that could meet this need and potentially be accessable to all classrooms and laboratories on campus. Teachers needing access to and maintenance for these new computing devices and support software could be spared the necessity of coping with technical interfacing detail by subscribing to these services provided by the Center through the Microsupport Net. The system described herein can serve as a model for others to evaluate.

PHYSICAL FACILITIES

An experimental, high-speed, recirculating microsupport net has been constructed with NSF support which employs synchronous time domain multiplexing. The network was designed to provide communications facilities for access to storage and support devices needed in the experimental application of microcomputer systems for experiment control and data acquisition. This same access facility can also be applied to downloading of assignments or programs to personal computers. Saving results for later evaluation by teachers completes the activity cycle.

The line that presently connects eight buildings on campus supports modified packet switching from station to station. It also concurrently supports the equivalent of hard-wired connections from terminals of other devices to a central station. An operational test phase has been completed, and remote testing is presently being pursued with full operation expected by spring 1979. Cables have been pulled to other parts of campus in anticipation of a steady growth in the length of the line and the number of associated connections. Connection costs of $25 to $35 per month are presently estimated to recover capital costs of support equipment and a personal computer for each connection.

The coaxial line operates at 4.632 megahertz with 256 bytes or slots of recirculating information as indicated in Figure 1. It has a theoretical capacity of two hundred fifty four 12,000 bits per second, one-way conversations between stations dedicated to specific byte positions or slots. The Microsupport Net is serviced by four types of stations: a centrally communicating station (CCS), a central station (CS), a general purpose station (GPS), and a single master station (MS).

Functionally, the MS provides master clocking, error logging, and automatic shut down of the Microsupport Net in case of reliability problems. A GPS is capable of communicating with another GPS at a 12,000 bits per second rate. A CS acts as a central facility for a CCS which in turn is dedicated to servicing up to eight serial terminals operating at rates up to 1200 bits per second. See Figure 2 for a sample configuration of the Microsupport Net.

Each station contains a battery-backed repeater for detecting the signal, repowering the signal, and shifting the data. In addition the GPS and CS contain an electronic hardware interface unit and a single board microcomputer (SBC). The interface units automatically handle all time critical line protocol and error handling functions in 0.3 microseconds. All standard slower functions are implemented in software through interrupts to the SBC. The CCS contains a reduced electronic hardware interface and eight universal asynchronous receiver transmitter UARTs. These UARTs share a single send byte (slot) and a single receive slot

---

This project was supported in part by NSF Grant No. SER76-18172.

which are time multiplexed. Thus, serial terminals, a mode readily available in personal computers, connected to a CCS have access to auxilliary file storage and computational capabilities associated with its corresponding CS. These in turn can be connected to even larger central mainframe computer facilities.

SUPPORT MODES OF OPERATION

A major use of the Microsupport Net is to permit relatively small, single-board computers or personal computers to monitor and/or collect experimental data without having to record them on site. This eliminates the need to chart, log, digitize, keypunch, etc., data for subsequent analysis. On-line processing of collected data for experiment control is also possible within time limitations.

Another major educational use permits the downloading of selected programs from the shared storage facilities on the Microsupport Net to a student's personal computer for subsequent stand-alone use. Results could then be logged back to the shared storage facilities for later access and evaluation by the teacher. This is not a time-sharing mode of computing, rather a distributed hierarchical form of computing which can more economically match computing resources to the needs of the person or problem. With the reality of low cost, personal computers, this access to shared resources through a Microsupport Net can amplify the usefulness and effectiveness of personal computers.

Addition of telephone dialup ports to CCSs would extend these capabilities to 1) collecting experimental data from remote sites without media conversion problems and 2) student's using their own personal computers at home or in their study rooms to fetch assignments from the Microsupport Net, work them on a stand-alone basis, and transmit results back through the Microsupport Net for subsequent evaluation.

RESULTS TO DATE

Two central stations presently connect eight personal computers through two centrally communicating stations. Each personal computer is located in a different building. Additional connections will be in place by June 1979 supported by two more CSs and CCSs. Presently a PDP 11/34 with UNIX operating system can handle about 6 to 8 simultaneous user requests without serious degredation of response times. Even so, with the mode of operation expected to be one in which many users need only occasional access to the Microsupport Net, this response time is not seen as a serious limitation.

Departments presently connected to the Microsupport Net at ISU include:

Electrical Engineering
Chemical Engineering
Industrial Engineering
Material Science Engineering
Engineering Mechanics
Mechanical Engineering
Computation Center
Cyclone Computer Laboratory

Additional connections planned for 1979 include:

Aerospace Engineering
Civil Engineering
Home Economics College
Veterinary College
Computer Science
Statistics
Psychology
Mathematics
Industrial Administration

Unsolicited inquires of interest for connection to date include:

Agricultural Engineering
Industrial Education
Animal Science
Physics
Entomology
Biochemistry

FUTURE PLANS

Additional microcomputer support services will be available on the PDP 11/34 for users needing access to assemblers and compilers for downloading of object code without media conversion problems or access to on-site development systems.

A series of service short courses and seminars will be continued by Center staff on a regular basis. User seminars on gaining access to the Microsupport Net, effectively using the resources, and selecting equipment compatible for connection to the Microsupport Net were initiated in January 1979 and will be continued on a regular basis.

Additional results and creative applications from shared experiences over the next few years should be numerous. The most significant problem forseen at this time is controlling an orderly and reliable pattern of growth to meet the exploding demand for personal access to Microsupport Net.

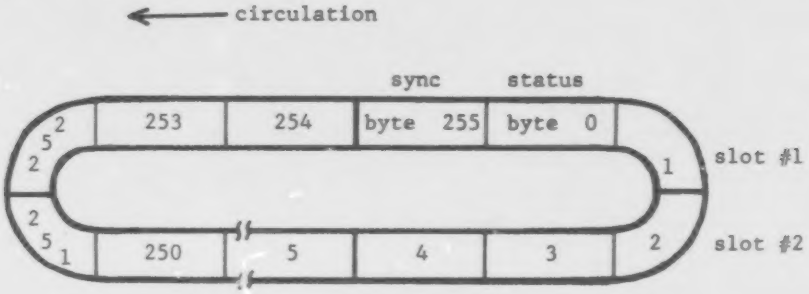

circulation
sync
status
253
254
byte 255
byte 0
252
1
slot #1
251
250
5
4
3
2
slot #2

Figure 1. Recirculating byte.

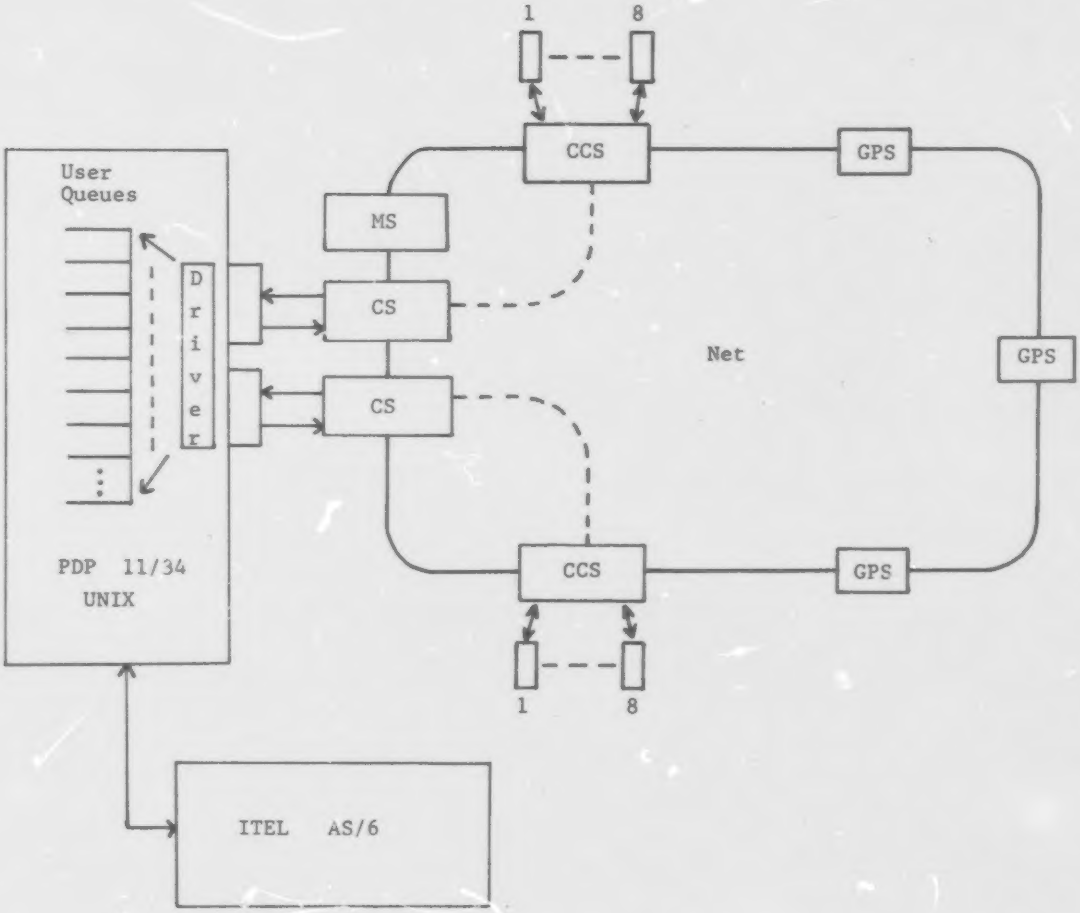

1
8
CCS
GPS
User Queues
MS
Driver
CS
CS
Net
GPS
PDP 11/34
UNIX
CCS
GPS
1
8
ITEL AS/6

Figure 2. Microsupport net configuration.

USING GRAPHICS FOR CAI WITH A MINICOMPUTER

Robert T. Ward*, Dept. of Physics
Martin Rudnick, Dept. of Curriculum
and Instruction
University of Northern Iowa
Cedar Falls, Iowa 50613
(319-273-6280)

Technological advances, primarily in the design of terminals, have made higher resolution graphics available at steadily decreasing costs. Several authors have described the benefits that graphics offers for computer-assisted instruction[1], but the most widely reported examples of actual use of such have required large or dedicated computers or system-special hardware (e.g., PLATO[2], TICCIT[3], Physics Computer Development Project at University of California, Irvine[4].) This paper reports on an experience developing CAI materials with graphics using Tektronix 4025 terminals run by a Hewlett-Packard 2000 minicomputer.

The University of Northern Iowa received support from National Science Foundation under the Comprehensive Assistance to Undergraduate Science Education program to develop an activity-based introductory science course for nonscience majors. The goal of the course is to enable students with little prior experience or interest in science to actively experience selected phenomena and concepts of science. As one of the general education courses at a medium-sized university, the course will serve relatively large numbers of students while using only limited faculty resources. The students to be served usually have neither the confidence nor the motivation to carry out analysis of problems or concepts without continual assistance and support. On the other hand, the concepts dealt with are fairly simple. These conditions suggested that it would be beneficial to include CAI materials of several types: tutorials designed to include extensive text, assistance, and remedial branching; programs which simulate complex natural phenomena; and programs that assist students in computations and analysis of experimental results.

Also due to the background of the students and the level and purpose of the course, we wished to avoid using abstract symbols and complex mathematical descriptions. We felt it important to be able to relate concepts directly to phenomena and concrete experiences. For this reason we needed high resolution graphics that would provide clear and detailed illustrations as an integral part of the CAI materials.

Another design limitation on our instructional materials was the available computer resources of the University of Northern Iowa. UNI has an HP2000 computer serving about 53 terminals on campus and in neighboring public schools. Available software includes the Instructional Management Facility (IDF) (which also keeps records on student utilization) and two CAI authoring languages, the Instructional Dialogue Facility (IDF), and the Course Writing Facility (CWF). Prior to our work, considerable use had been made of the IMF for lessons in IDF, but no prior work had been done using CWF. CWF is similar to and compatible with IBM Coursewriter III.

A. Bork and others have made a good case for using microcomputers to present CAI materials. There were several reasons why instead we decided to work with a system using the largest computer which we had available to us. First of all, we had had little prior experience with CAI so the available author languages offered structure, documentation, and guidance. Second, we planned to use complex lessons, including sophisticated answering processing techniques, branching, and graphics. Third, we wanted to use the management, record, and response keeping facilities for formative evaluation of our materials as well as for student evaluation. Finally, by using the central computer we had access to numerous terminals for non-graphic applications.

The National Science Foundation support allowed us to acquire several graphics terminals specifically for this project. After some comparison we decided on the raster scan type that permits selective erase and scrolling of graphics and text. This is especially useful for CAI where one may wish to change graphics or text without completely erasing the screen and redrawing the figures.

---

* will deliver talk and handle correspondence.

We acquired three Tektronix 4025 terminals as well as a hard copy unit for use in our development work. In order to input complex figures we also purchased a graphics digitizer (Graf Pen-6).

At present most of our CAI materials are being written using CWF. Many argue that it is best to write CAI directly in a computer language such as APL or BASIC, but the only language available to us was BASIC. Because none of the staff had had much experience in developing CAI, it was desirable to use the structure offered by an author language and the available student management facilities and record keeping system. The CWF author language is more difficult to use than IDF, especially initially, but it is superior in its flexibility and in its ability to branch to BASIC at any point in the lesson. BASIC programs are used to draw most of our graphics, as well as for portions of the lesson which involve calculations or other analyses that are not possible in CWF. CWF not only allows the lesson to transfer to and from a BASIC program but also permits a modification of a student's response data in either the BASIC or CWF portion of the lesson. As a result, even programs that are not tutorial and thus are less suitable for CWF have been written in BASIC and then chained to from an introductory section written in CWF so that the student management and record keeping systems can still be used.

Our experience with CWF has not been without difficulties. The instructional manuals seem inadequate. There is no built-in system to keep records on which specific questions in a lesson a student has attempted or answered correctly. Also CWF does not have established procedures for producing graphics on our terminals.

We are still struggling to gain a better understanding of CWF, but we have been able to develop new software to supplement it in those areas where we noted inadequacies. To keep a complete record of each student's achievement we use the buffers, registers, counters, and switches that CWF establishes for each student. Using software developed by our Academic Computing Service the instructor can obtain information on how far the student has progressed, where he or she has been branched, and which questions were answered correctly. Even if the student signs off and returns to repeat some questions, the records correctly reflect the progress. All of this entails a large number of commands in the CWF lesson, but this is greatly simplified through the use of macros. (These are similar to subroutines that once written out in a lesson can be called upon by just one line of commands and varied slightly as needed in each question, correct answer, wrong answer, etc.)

The other supplement to CWF which we have developed is a system to store the student responses. Just before a student completes a given question, the program chains to a BASIC program which stores all of the student's responses to that question (correct answers, wrong answers, and unexpected answers) in a student response file, labeled with the time entered, the student's name, and question number. We can then call up all answers to a single question or all the responses of one student. Since these responses are stored in chronological order one can also get from this file a complete chart of a student's responses, with all branches, repeats, time to do any given section, number of trials, etc. This storing of student input does slow down the response time of the system. Therefore it may not be worth using after the formative evaluation of the lesson has been completed.

We have developed a system to produce, store, and present graphic materials as needed for the CAI lessons. The software for this system was written by the UNI Academic Computer Service under the direction of Kenneth Cox. The form is a graphic author language which enhances the graphic generation capabilities of the Tektronix 4025 terminals. It includes commands for the drawing of straight lines, regular polygons, and arcs; for the scaling, translating, and rotating of such figures; for specifying line types; and for positioning strings of characters. Sets of such commands are combined into graphic generation programs and the coordinates of the resulting figures are stored in files. The digitizer may also be used to input the coordinates of more complex figures. One may then transform and combine these figures. In this way it is convenient to manipulate various elements until a suitable figure is obtained. The graphics generation system has been written in BASIC and, with slight revisions, should be readily transportable.

In addition to the large effort involved in producing CAI materials, at least three-quarters of our time has been put into developing the software systems described above. As a result we have produced and used with students only five CAI lessons to date: an introductory dialogue, two tutorial lessons, an experimental data checking and recording lesson, and a simulation.

As mentioned earlier, the students for whom we are developing materials are nonscience majors, with no prior experience with computers and often considerable fear of them. Our first lesson is an introduction to the interactive CAI procedures: how to sign on, input responses, make corrections, and sign off. By giving students practice on the terminal (including playing Hangman and responding to a knock-knock joke) when there is no science content involved, they find the introduction enjoyable and their fears allayed. During their first session on the terminal we have assistance available.

The first content lesson is on grouping and classification of objects. The topic is introduced by a laboratory activity on grouping tree leaves. Then each student is required to take the CAI lesson on classification which introduces some terminology and the concept of a dichotomous classification system. The concept of a dichotomous key is then used in a laboratory and field trip activity to identify local tree leaves. Thus the CAI lesson is closely tied to activities and real phenomena. The availability of computer graphics allows the CAI lesson to present a set of objects (e.g. simple geometrical figures) and to require the student to make decisions about classifying them. Using graphics the computer can then display the student's choice and require further decisions so that the entire classification process is carried out. Here the selective erase aspects of the graphics terminals greatly simplify and speed up the procedure.

A variation on computer-assisted testing is used in an activity on density. The students were each assigned several pieces of minerals and required to measure the weight and volume. (This was to be done outside of class time). In order to check their results and detect major measurement errors, students were to input sample numbers and results into the computer which informed them if the values were sufficiently close to standard results. This computer program also kept a record of the number of attempts and produced a listing of all student results. By presenting in class the graph of weight against volume for all the student results, we introduced the concept of density as a characteristic property of a given mineral.

A more traditional type of CAI is a tutorial lesson entitled LIF, which is a discussion of the changes of and transfer of energy involved in the lifting of a barbell by a weight lifter. Here graphics are used to display figures which illustrate the process and to present force diagrams. This lesson is given to the students after they have done a lab activity on an inclined air track, a phenomenon in some ways analogous to the weight lifter and barbell. The dialogue assumes that the student has previously been introduced to the concepts of kinetic and potential energy. It requires the student to use these concepts in a qualitative discussion of energy changes. There are extensive remedial branches for the student who has difficulty applying the concepts.

The simulation which we have produced is a modification of an available program written in BASIC which concerns the ecological relationships between animals in a grassland. In this lesson, Grassland[5], the student choses initial populations for cattle, songbirds, and hawks. The program then computes values for the populations of these animals, plus rodents and grasshoppers, over a 15 year interval. The changing populations are displayed in graphs in order to clarify the results and relationships for non-mathematically skilled students. Questions are used to insure that students read and interpret the graphs correctly.

Several more CAI lessons or materials are in the development or planning stages. Because we have been developing the system and procedures, the process of producing materials has required continual and extensive interaction between the instructor, an instructional design specialist, programmers (working in BASIC), and CAI writers (working in CWF). We are developing a more systematic and structured procedure so that, with consultant assistance from an instructional designer, the instructor (content and pedagogy specialist) can specify the text, questions, responses, graphic material, and branching strategies. Then a CAI programmer (undergraduate student assistance or paraprofessional) can code the lesson into CWF. The procedure will enable the instructor to specify the format of the text and graphics and the way in which they will be modified based on the responses of the student.

Our experiences to date have convinced us that a system can be developed that will make it possible to use computer-assisted instruction with graphics without an unreasonable investment of time by the instructors involved. It will take a significant effort to perfect such procedures, but the promised benefits in cost and time efficiency are great.

References:

(1) Alfred Bork, "Learning Via Computer Graphics," NATO Advanced Study Institute, Louvain-la-Neuve, Belgium, July, 1976.
(2) William C. Norris, AEDS Monitor, Vol. 17, n 4-6, p. 7 (Oct/Nov/Dec 1978).
(3) Allen L. Lammond, Science, Vol. 176, n 4039, p. 1110 (6, June, 1972).
(4) Alfred Bork, Educom, Vol. 10, n 3, p. 14 (Fall 1975).
(5) Based on "GRAZE Ecology Simulation", Michael Chester (Hewlett Packard Company, 1973)

A Computer-Assisted Instruction System for the Blind and Visually Impaired*

1979

William L. Ballenger, Ph.D.
School of Education
North Carolina State University
P. O. Box 5096
Raleigh, NC 27650
919/737-3127

Project Purpose

The general purpose of the project was to reengineer the human access used in computer technology and use the techniques to develop a computer-based instruction system to augment rehabilitation training programs for the blind and visually impaired. Specifically, the technological developments are to be used to (1) facilitate greater employment of the blind and visually impaired as a result of job reengineering in computer-related activities, (2) upgrade the overall quality and quantity of training methods for the blind and visually impaired, and (3) increase the accessibility of blind and visually impaired to a wider range of vocational and technical jobs.

Primary Design Objective

The primary design objective was to develop a computer-assisted instruction (CAI) system which would be compatible with the information acquisition abilities of the blind and visually impaired. The term 'system' is herein defined to include instructional lesson production and administration. Lesson production is limited to the translation and assembly of author-prepared instructional material for CAI application. Lesson administration includes the activities of a computer-controlled communication of instructional information and record keeping of all transactions between the user and CAI system.

For computer-assisted instruction applications there are several media alternatives for information input to the learner without dependence upon normal vision. These alternatives include large print (visual), braille (tactile), the OPTACON reading system (tactile), and auding (auditory). The criteria used to evaluate each medium for information input include the percentage of the blind and visually impaired population that can use the medium, comprehension of language presented by the medium, the words-per-minute information transfer of the medium, and learner skill development for using the medium. The criteria used to evaluate each medium for CAI application are the following: the technological feasibility, availability and cost of equipment, the relative difficulty of human activity in using the medium, and the relative software development task for recording and presenting the instructional material for use through the medium.

Table 1 summarizes the factors for each of the four media. It appears that speech is the preferable alternative given the instructional and technological considerations. Although audio response units are not the least costly, their technical integration into a CAI user facility is one of the most elegant in terms of simplicity and reliability. Audio response units are constructed largely of reliable solid state electronics. The units are plug compatible with existing 30 cps terminals: that is, the hardware implementation of CAI with synthesized speech is simply a matter of plugging the unmodified hardware units together. This implementation requires the development of supporting software or modification of existing software to control the audio response unit. Of all the media alternatives, speech is the most costly in terms of system development and lesson production. Considering the population that can be served by the medium, the expected instructional effectiveness of the medium, the technological simplicity, and expected reliability of the hardware, the cost of development and production of a Computer-Assisted Instruction with Computer Output Speech (CAI/COS) system can be well justified (Ballenger, 1978).

Project Approach and Accomplishments

Computer-Assisted Instruction with Computer Output Speech is built around two developments: 1) the hardware to interface the blind and visually impaired

---

*This research has been supported by a grant from Rehabilitation Services Administration, Department of Health, Education and Welfare, Grant #6532617 administered by the North Carolina Department of Human Resources Division of Services for the Blind and North Carolina State University.

FACTOR	MEDIA				PREFERENCE
	LARGE PRINT	BRAILLE	OPTACON	SPEECH	
ESTIMATED USER POPULATION	75% OF TOTAL	2% OF TOTAL	.1% OF TOTAL	98% OF TOTAL	SPEECH
LANGUAGE COMPREHENSION	EQUIVALENT FOR ALL MEDIA AT AVERAGE WORD DECODING RATE				
AVERAGE WORD DECODING RATE	70 WORDS PER MINUTE	100 WORDS PER MINUTE	45 WORDS PER MINUTE	175 WORDS PER MINUTE	SPEECH
DECODING SKILL DEVELOPMENT AGE	AGE 12	AGE 15	AGE 15	AGE 10	SPEECH
USER TERMINAL EQUIPMENT COST	$3,000 TO $12,000	$4,000 TO $14,000	$4,000 TO $6,000	$5,000 TO $9,000	LARGE PRINT
USER ERGONOMIC ACTIVITY	MAJOR	MODERATE	MODERATE	MINOR	SPEECH
SOFTWARE DEVELOPMENT	MINOR	MAJOR	MODERATE	EXTENSIVE	LARGE PRINT

COMPARISON OF MEDIA ALTERNATIVES

TABLE 1

with an unmodified computer and 2) computer software to produce lessons and manage an instructional dialogue. The hardware, or CAI/COS station, solved the problem of the human/computer access for the visually impaired by replacing the visual display with electronically synthesized speech output. The instructional delivery software was designed to maintain a keyboard/speech dialogue between the computer and student.

The project accomplishments can be discussed at several levels. In terms of the overall project objectives, an instructional system for augmenting Rehabilitation Services Training Programs was designed, developed, tested and is now available for use by training programs that wish to connect to the North Carolina Educational Computing Service network. The North Carolina Educational Computing Service is an operational unit of the University of North Carolina General Administration and is located at Research Triangle Park N.C. Computing services for the network are leased from the Triangle Universities Computation Center, (TUCC), a non-profit educational computer corporation, at Research Triangle Park N. C.

More specific accomplishments can be discussed in terms of technological and instructional developments that have utility for job reengineering and rehabilitation training. The development of the Computer-Assisted Instruction with Computer Output Speech system is composed of computer hardware, software, and the instructionware.

Hardware:

Hardware components fall into two categories, remote computer and CAI/COS station. The computer used by the project and for future applications is an unmodified Hewlett-Packard 2000 System minicomputer (Hewlett-Packard, 1975 a). The CAI/COS station consists of a terminal, an electronic speech synthesizer, an audio speaker, and telephone communications equipment.

Any terminal allowing two-way communication, i.e., containing both a keyboard and a display, and that can send and receive data at 30 characters per second will suffice. The electronic speech synthesizer used in the CAI/COS station is the VOTRAX ML-1E manufactured by the Vocal Interface Division, Federal Screw Works, Troy, Michigan. The ML-1E is driven by computer-generated ASCII codes representing 84 phoneme elements of the spoken English language. The ML-1E is installed by simple cable connection (EIA RS-232 plug) between the CAI/COS station terminal and the telephone communications equipment (modem or data set). This telephone link frees the location of the terminal and allows it to be located wherever there are students who need it. Given the rate of transmission (30 characters per second) any voice-grade telephone line will suffice. All of the components in the CAI/COS station are standard commercial items. It was not necessary to use any custom-designed electronic equipment for the development of the CAI/COS station.

The audio speaker used with the CAI/COS station can be any small, low-impedance speaker. The Project developed a sound system by adding switches, audio jacks, and an audio mixing circuit to a standard speaker. The student is thus able to use either the speaker or headphones. The user may also record on cassette tape the speech output from the lesson mixed with whatever comments he/she wishes to voice over a headphone-mounted microphone. Additionally, a proctor can monitor a student's session either by listening to the speaker or by plugging his/her own headphones into the speaker box. The proctor may also tape record comments using a headphone-mounted microphone.

Software:

Collectively the computer software produced by the project for CAI/COS falls into three groups: lesson production, speech translation, and lesson administration. A principal component of the lesson production system is the Instructional Dialogue Facility (IDF), a group of commercially available computer programs written by the Hewlett-Packard Company for use with their 2000 computer system (Hewlett-Packard, 1975 b & c). The Hewlett-Packard software and computer system were chosen for this project because they were both locally available and in wide use throughout the country. Additionally, the Hewlett-Packard systems exhibit broad CAI capabilities and exceptional reliability.

The IDF includes a computer program which enables authors possessing no computer programming skills to produce a CAI lesson by entering lesson components on a computer terminal in response to prompts and messages issued by the computer. The program also allows the author to specify lesson options such as the amount of time allowed for student response, the number of attempts allowed per question, and the type of data to be gathered by the CAI/COS operating system during the lesson presentation. The CAI lesson resulting from an author's use of the IDF is a formatted computer file, resident on the computer disk storage system. This file is compatible with other Hewlett-Packard Company lesson delivery software for conventional printout on paper or terminal video screen.

After generation of a CAI lesson, Project-developed software performs several functions in translating the IDF formatted CAI lesson into a speech version compatible with the CAI/COS delivery system. Editing facilities are provided for correction of any mistakes made during CAI lesson entry and for changes necessary to observe several lesson-writing conventions that yield a more intelligible speech version. Lessons previously written for conventional CAI delivery can be edited before translation for compatibility with vocalization rules. A computer program written by Project staff then scans the edited CAI lesson text for any new words whose speech synthesis codes are not stored in a computer-based lexicon of speech coded words.

New words identified by the text scan may be translated into speech synthesis codes by one of two means. One is a computer program developed by the Naval Research Laboratory (Elovitz et al., 1976). This translation program is based upon a set of letter-to-sound rules. The English language is composed of distinct sounds called phonemes. As the letter-to-sound rules are applied to a word, the phoneme components are determined and represented by a computer code. The second is VPACE, or VOTRAX Phoneme ASCII Code Editor, a program developed by the project to both manually generate and modify coded computer controls for individual phoneme-based word and phrase sounds, and to edit and improve the phoneme codes automatically generated by the NRL program. Newly translated and edited words are added to the lexicon of speech coded words.

Other project software named LESA (LESson Assembly) then assembles a CAI/COS lesson word by word, substituting synthesis coded from the lexicon for the orthographic words of the conventional CAI lesson. Finally, the assembled CAI/COS lesson may be manually edited to accommodate any speech code changes indicated by the context in which a word or words appears. Individual phonemes of a word or statement may be replaced, inserted, or deleted as needed to improve clarity of the speech presentation.

The computer-based lexicon produced as a result of lesson generation is a residual resource which can be used in subsequent lesson development. Once translated, words can be used in any future lesson, requiring only occasional phonetic editing to reflect context. The stored lexicon method of lesson production was chosen over real-time translation because it provides maximum control over speech quality. A series of CAI/COS evaluation lessons of approximately 200,000 words (22 hours of speech at 150 words per minute) is fully supported by a 2,000 word lexicon.

Lessons generated and translated are compatible with a Project modified version of Hewlett-Packard's Instructional Management Facility (IMF), a system of computer programs which manages system security, lesson delivery, and the collection of system and student performance data (Hewlett-Packard, 1975 d, e & f). The project software is herein referred to as IMF/COS.

IMF/COS limits access to those students who have been entered into an IMF/COS student file and enrolled in the course of which the requested CAI/COS lesson is a part. Upon calling the IMF/COS system, the following sequence of events are prompted by vocalized request from the computer. The system first requests the student to give an identification. After the student identifies himself/herself to the system by typing his/her unique identification number and first name, the IMF/COS asks for name verification by vocally spelling the student's last name. Next the system asks for the name of the desired lesson and verifies the lesson name by spelling out each letter. Lesson delivery begins after the IMF/COS determines that the student is authorized to take that lesson and has not already completed it. If the student has begun the lesson in a previous session, the IMF/COS begins lesson delivery with the appropriate section of the lesson. Lesson delivery continues in a conversational order with the system reading the instructions, questions, and responses until the student has completed or elects to terminate the session. Several commands allow the student to obtain his/her score, to listen to a section repeated, or to hear hints if the author has provided them.

Instructionware:

The instructionware for the project can be grouped into three types: 1) keyboard training, 2) speech intelligibility and student readiness testing, and 3) vocational/technical instruction. The first two types were developed as functional support of the CAI/COS system. The three types of instructionware were written for, and as an example of, the project's objective.

The CAI/COS station assembled for the project employed a small minicomputer ($8,000) as the terminal component. This level of terminal is not required of the CAI/COS station and may be functionally replaced with a 30 cps keyboard terminal for all applications except the keyboard training program defined below. The keyboard training program was developed to train blind and visually impaired persons to use the typewriter-like keyboard of the CAI/COS station. Initially, the student is taught the key labels and locations on the keyboard with a cassette tape lesson. The CAI/COS station minicomputer was programmed so that when the student strikes a key, the label is annunciated by the synthesizer. Using this keyboard program and the cassette tapes, the student can become familiar with the keyboard and develop the necessary typing skills to communicate with the CAI/COS instructional programs.

A second support program was developed: 1) to test the student's ability to hear and understand the information being presented by the speech synthesizer, 2) to test the student's ability to respond through the keyboard to questions asked by the computer, and 3) to confirm the student's readiness to use the CAI/COS system for instructional application. This CAI/COS support program resides in the HP/2000 CAI/COS Library, is accessible by all users on the NCECS network, and can be used with a minimum terminal CAI/COS station configuration.

The vocational/technical instructional programs written by the project staff members were to serve two functions: 1) to provide a test for CAI/COS application and 2) to provide supplemental instruction in basic accounting. The accounting course is composed of twelve lessons. Each lesson has from 20 to 30 sections composed of text, question, response to correct answer, responses to incorrect answers, response to unexpected answers, and a failure message.

The concept of course development is to build a CAI/COS Library of instructional programs that can be used by training programs in universities, colleges, technical institutes, and rehabilitation centers. As the system is used to support various training needs of the blind and visually impaired, the library will continue to collect more and more instructional material to support the efforts of rehabilitation.

The objectives of this project were to reengineer the human/computer access for use by the blind and visually impaired and to develop an instructional support system for Rehabilitation Services Training Programs. The number of rehabilitation students involved with the testing and evaluation of the keyboard/speech computer interface and instructional programs was small. For development purposes a single Rehabilitation Services Training Program at Pitt Technical Institute in Greenville, North Carolina was chosen for a student base. During the project student involvement ranged from learning the terminal keyboard by using cassette tape instruction and computer speech feedback, to using the instructional programs as a concurrent supplement to their regular vocational and technical studies. During the project at Pitt Technical Institute there was not a sufficient population of blind and visually impaired students in various vocational and technical programs to develop an evaluation of the CAI/COS system.

Project Implications

The technology of electronically synthesized computer output speech holds great promise in job reengineering for the blind and visually impaired. Many jobs developed around the use of computer data base query (information acquisition), such as those in employment counseling, reservations, inventory control, and library science, can now be made available to the handicapped, reducing the social and economic costs of their unemployment or underemployment. Information Service Expediting is a broad occupational category which includes the job areas mentioned above. In 1975 the Internal Revenue Service, the Social Security Administration, and the Civil Service Administration of the United States Government employed approximately 275 blind information service expediters (Vieceli, 1975). Implementation of computer speech systems in the private sector should mean greater quality employment opportunities for the visually impaired. In addition to private sector employment, the use of computers via telecommunications has a vast potential for the development of home industries for the blind and visually impaired. Computer terminals located in the home could provide a work station for individuals trained in any of the several fields of applications that use computer-based information and/or processing.

CAI/COS as an instructional medium has numerous applications and extensions. These include use at the elementary and secondary school levels, diagnosis and correction of certain reading difficulties, development of an inexpensive CAI/COS station, and telecommunications stratagies for home use and on the job training.

Studies by the U. S. Army (Sticht, 1971) and the U. S. Air Force (Sticht et al., 1974) have indicated that, in some situations, learning by listening is as effective, or more effective, than learning by reading. Military applications of speech instruction have been directed primarily at men with relatively low reading skills. CAI/COS may be useful in imparting some knowledge at the lower grade levels where students have not yet acquired adequate braille or low vision reading skills. The potential of CAI/COS in the instruction of the less motivated student at any age level should be explored.

The technologically feasible combination of computer-output speech and an inexpensive touch-tone encoder terminal (currently available at under $200) implies great accessibility to conversational instruction or computer data base query for anyone having access to a telephone. The homebound, geographically dispersed agents of an organization, or any print handicapped individual could benefit from readily available, low cost conversational CAI/COS instruction or information delivery. In addition, this technology may be of great value to employers as an auxiliary aid to the sensory impaired in meeting the mandated regulation: Subpart F, 84.52 (d) of Section 504 of the Rehabilitation Act of 1973.

References

Ballenger, W.L., Objective and alternatives for a computer assisted instruction system for the visually handicapped. Unpublished project report, North Carolina State University, Raleigh 1978.

Elovitz, H.S., Johnson, R.W., McHugh, A., & Shore, J.E. Automatic translation of English text to phonetics by means of letter-to-sound rules (NRL Report 7948). Washington, D.C.: Naval Research Laboratory, 1976.

Hewlett-Packard Co. HP 2000 series system and subsystems data. Palo Alto, Ca.: Author, Mar., 1975. (a)

Hewlett-Packard Co. HP 2000 instructional dialogue facility author's manual (Part No. 22691-90003). Cupertino, Ca.: Sept., 1975. (b)

Hewlett-Packard Co. HP 2000 instructional dialogue facility proctor's manual (Part No. 22691-90001). Cupertino, Ca.: Sept., 1975. (c)

Hewlett-Packard Co. HP 2000 instructional management facility proctor's manual (Part No. 22690-90001). Cupertino, Ca.: Sept. 1975. (d)

Hewlett-Packard Co. HP 2000 instructional management facility course developers' manual (Part No. 22691-90002). Cupertino, Ca.: Sept., 1975. (e)

Hewlett-Packard Co. HP 2000 instructional management facility system manager's reference manual (Part No. 22690-90002). Cupertino, Ca.: Sept., 1975. (f)

Sticht, T. G. Learning by listening in relation to aptitude, reading, and rate-controlled speech (HumRRO Tech. Rep. 71-5). Alexandria, Va.: Human Resources Research Organization, 1971.

Sticht, T.G., Beck, L.J., Hauke, R.N., Kleiman, G.M., & James, J.H. Auding and reading: A developmental model (HumRRO Tech. Rep. 74-11). Alexandria, Va.: Human Resources Research Organization, 1974.

Vieceli, L., Guidelines for the selection, training, and placement of blind persons in information service expediting. Carbondale, Ill.: Southern Illinois University, Rehabilitation Institute, 1975.

SPEDINFO:

A Decision-Making Support Feature For The PLATO System

John Patrick Cronin
Peter J. Rizza, Jr.
Control Data Education Company

This paper addresses the development of a PLATO feature to assist different professional school personnel in implementing recent federal legislation {P.L. 94-142, The Education for All Handicapped Children Act} as well as the variety of state mandates which may or may not be consistent with the federal law.

The implementation of this law is one of the largest problems facing educators in many years. Many school districts, attempting to avoid legal punishments, have simply refused to recognize and serve certain handicapped children. This fact is documented by U.S. Department of Health, Education, and Welfare reports citing that there are some 8-10 million handicapped children, ages 5-18, eligible to attend the nation's schools, but as of fall 1978, only some 3.9 million are actually enrolled.

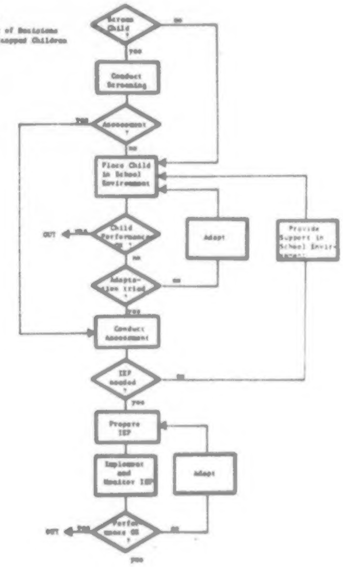
The law requires that a legal document be written for each student who receives special education service. This document, called the Individualized Education Plan {IEP}, is the result of careful and systematic observation, testing, consultation, and interaction with the child, the parents, teachers, and any other appropriate persons in determining the best educational program for each child. This IEP is the culmination of these planning efforts and must be agreed to by all parties involved before it is enacted. {If any party does not agree, the law spells out ways to settle the issues through other legal avenues.}

Public Law 94-142 and other legislation recently enacted to demand formal procedures which permit the clinical assessment of the special needs and strengths of children. In general, this involves {a} screening, {b} assessment, and {c} instructional program planning based upon {a} and {b}. The adjustment of the school environment and instructional practices to the condition of the student as implied in the IEP can be viewed as a series of decisions to be made by educators, parents, and students. The sequence of decisions is shown in Figure 1.

Children new to a school district enter Figure 1 at the top. If there are reasons to conduct screening, or the child is at an age where all children are screened, a suitable procedure is carried out to determine needs for special programming. Where there is no screening of the student, or where screening does not indicate a need for assessment, the child is placed directly in the classroom setting. Performance monitoring identifies potential problems and leads to adjustment of the learning environment. If such adjustments prove to be unsuccessful, children may be referred to assessment. These assessments indicate student strengths and weaknesses and the need for a detailed IEP. In some cases, assessment may clearly point to possible environmental adjustments without detailed programming.

For Children with special needs, educators, parents, and the students themselves engage in collective determination of the IEP. This Plan is implemented and monitored, and adjusted in those instances where the child does not make desired developmental progress. Successful levels of performance lead to exit from the procedures pictured in Figure 1

Figure 1.
Structure of Decisions For Handicapped Children


Screen Child ?
no
yes
Conduct Screening
yes
Assessment ?
no
Place Child in School Environment
OUT
yes
Child Performance OK ?
no
Adapt
Provide Support in School Environment
Adaptation tried ?
no
yes
Conduct Assessment
IEP needed ?
no
yes
Prepare IEP
Implement and Monitor IEP
Adapt
OUT
yes
Performance OK ?
no
yes

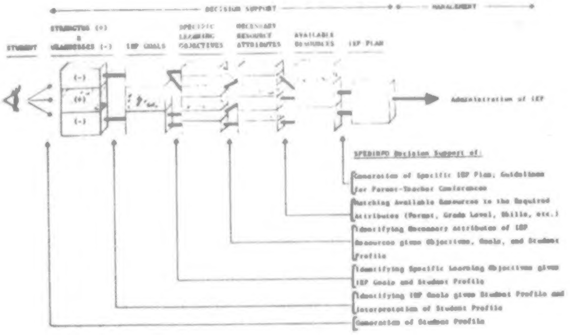

DECISION SUPPORT
MANAGEMENT
STUDENT
STRENGTHS (+) & WEAKNESSES (-)
IEP GOALS
SPECIFIC LEARNING OBJECTIVES
NECESSARY RESOURCE ATTRIBUTES
AVAILABLE RESOURCES
IEP PLAN
(-)
(+)
(-)
Administration of IEP
SPEDIRPO Decision Support of:
Generation of Specific IEP Plan; Guidelines for Parent-Teacher Conferences
Matching Available Resources to the Required Attributes (Format, Grade Level, Skills, etc.)
Identifying Necessary Attributes of IEP Resources given Objectives, Goals, and Student Profile
Identifying Specific Learning Objectives given IEP Goals and Student Profile
Identifying IEP Goals given Student Profile and Interpretation of Student Profile
Generation of Student Profile

Figure 2. Overview of Decision Support in the SPEDIRPO System

What is SPEDINFO? SPEDINFO is a decision support system for educators which provides assistance in meeting the instructional needs of handicapped students. The design for the system is based upon reviews of literature in the field and expert opinion gathered across the United States.

The information base for SPEDINFO is composed of information that the decision makers must draw upon to arrive at workable instructional plans. These are:

1. The educational objectives of the school program and associated teaching/learning materials.
2. The personal psycho-social characteristics of the child and the educational strategies appropriate to them.

This information, which exists in general within the information based of the proposed system, is augmented by specific knowledge of the local environment within which decision making is being carried out. SPEDINFO is a decision support system that guides education professionals and parents in the analysis of the child's strengths and weaknesses. SPEDINFO assists in the diagnosis of the child in screening and assessment activities. Given locally derived measures on child behavior and development, SPEDINFO points to subject matters appropriate to the child's intellectual level. SPEDINFO also catalogs the local materials of instruction and teaching strategies so that the needs of the student can be matched in the educational programming available to him or her.

Because SPEDINFO is designed as a flexible support system, users can enter its logic at any point. Thus, the screening of four-year-olds and initial instructional programming can be accomodated by SPEDINFO as can referrals of children of all ages by teachers. Because SPEDINFO is based on educational objectives of a functional nature, it can serve as a foundation for the educational programs of all children. As the information base for the system is enlarged, higher levels of educational objectives can be entered so that the system can ultimately support long-range career planning for children of all ability levels.

Objectives and Materials

The academic foundation of SPEDINFO is a set of educational objectives that span the range of behaviors required of all functioning members of the society. These objectives are derived from analyses of the tasks of adult work and social life, and are seen as the minimal bases for intellectual development of all children. The objectives, in turn, are keyed to available instructional materials so that appropriate selections can be made in planning a child's IEP.

The prototype SPEDINFO is based on a choice among several alternative collections of educational objectives: Fountain Valley Curriculum; Council for Exceptional Children Career Educational Objectives; HELPS Project Material Index. In the proposed system the user may adopt whatever materials are available to him or her and through interaction with the system tailor the use of these materials to the needs of his or her students. No matter what the choice of objectives the system development tasks remain roughly the same:

a. Placing the Child in the Curriculum. The level at which the child first enters the curriculum is determined by a diagnostic testing procedure. Initial child data enable educators to select the appropriate level of the testing program. Children new to the school or those who appear to be misplaced in the curriculum are given a general achievement test followed by a specific placement test.
b. Selection of Instructional Materials. The first procedure identifies those objectives appropriate for the child. SPEDINFO then provides a catalog of instructional materials along with guidelines for teacher decision making. Later versions of the system incorporate procedures for classroom organization and scheduling of instruction.
c. Modification of Instruction. The logic pictured in Figure 1 indicates that the instructional environment of the classroom can be changed to adapt to the special needs of the student. This modification includes adjustment of curriculum materials, changes in instructional tactics, and other alterations in teaching practice or classroom organization. SPEDINFO assists in the adjustment of instruction by outlining observations teachers might make and the adjustments which might follow.

Not all instructional modification will be successful, however. In those cases where modifications have been tried unsuccessfully, SPEDINFO assists the teacher in making the referral decision which moves students into a formal psycho-social assessment.

Child Profile and Educational Strategies - IEP Writing

The second component of SPEDINFO has to do with the psychosocial development of the child--his or her special strengths and weakenesses--and the responses the school might make to further intellectual and social growth. Educators using this system component believe that the child has unique strengths and weaknesses which cannot be readily accommodated in the conventional instructional programs, a belief based on either screening information or classroom referral {as shown in Figure 1}. Using the decision support available in SPEDINFO, educators can carry out a thorough assessment of the child and can interpret assessment results in a manner which enables them to select appropriate instructional programs {IEPs}.

The decision support for this aspect of student analysis and programming is unique in that it provides for the legally mandated group decision making in IEP preparation. SPEDINFO assists in organizing child assessment data and instructional alternatives for the IEP writing exercise and helps educators and parents weight possible program decision consequences. The hard-copy feature allows users to prepare program control documents that can guide instruction and evaluation of child performance as well as assist parents in understanding the educational plan for their children.

Categories of student strengths and weaknesses are identified as are the instructional practices related to them. The IEP decision support also conforms to a district's practices, and SPEDINFO parallels the process of professional diagnosis and instructional prescription which can be replicated in other school settings:

a. Profiling Student Strengths and Weaknesses. Preparing educational plans for individual students {IEPs} is possible only is students can in some way be categorized prior to assignment to programs or instructional experiences. To be useful to practitioners, any such categories must conform to the ways teachers, administrators, and special educators currently view students. SPEDINFO's child profile is based on a set of such naturally occurring categories and contains the guidelines for translating assessment data into category membership for a given student. Users enter the decision support environment with assessment data and are assisted in developing a categorical strength-weakness profile for the child.

b. Choosing Instructional Strategies. The profile developed above is input to the IEP writing exercise along with those instructional strategies proven useful for the mainstreaming of special needs students. SPEDINFO dimensions these strategies so that they can be related to individual student needs. Such dimensions as staff requirements, classroom organization, and cost are examples of information SPEDINFO presents to students.

c. Preparing the IEP. The IEP writing exercise is based on the collected information about student strengths and weaknesses and the instructional strategies available to meet identified needs. These two problem components are presented in a decision support environment so that parents and educators can be guided in the IEP preparation. When an IEP has been agreed upon, SPEDINFO assists in formatting the program. Hardcopies of this format are provided to control classroom instruction and follow up evaluation of student performance.

Objective and Materials

What we have in the objectives and materials compoment of the SPEDINFO system is in reality two matrices of information: educational objectives and materials, and relationships between observed student needs and teaching tactics.

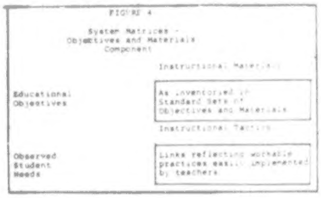


FIGURE 4

System Matrices -
Objectives and Materials
Component

Instructional Materials
Educational Objectives
As inventoried in Standard Sets of Objectives and Materials
Instructional Tactics
Observed Student Needs
Links reflecting workable practices easily implemented by teachers

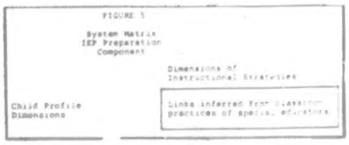


FIGURE 5

System Matrix
IEP Preparation
Component

Dimensions of Instructional Strategies
Child Profile Dimensions
Links inferred from classroom practices of special educators

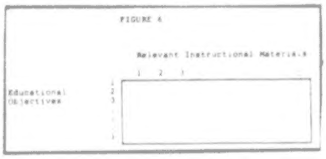


FIGURE 6

Relevant Instructional Materials
1 2 3
Educational Objectives
1
2
3

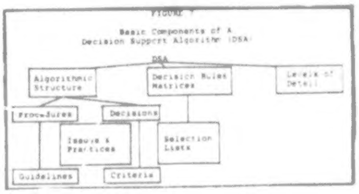


FIGURE 7

Basic Components of A
Decision Support Algorithm (DSA)

DSA
Algorithmic Structure
Decision Rules Matrices
Levels of Detail
Procedures
Decisions
Issues & Practices
Selection Lists
Guidelines
Criteria

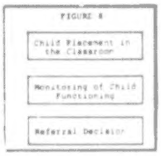
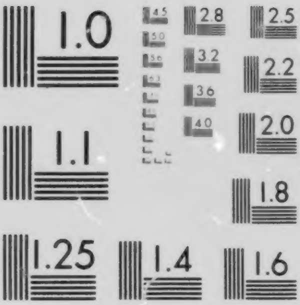


FIGURE 8

Child Placement in the Classroom
Monitoring of Child Functioning
Referral Decision


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3

MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

The first of the above matrices is determined by the functional curriculum selected for the prototype. That is, choice of the Fountain Valley curriculum, for example, provides the total information input for building the objectives-materials matrix. The second matrix requires classroom observations that are easily interpreted and can be used to suggest those teaching and classroom organization tactics most likely to meet observed student needs. This matrix must, of course, conform in significant ways to the more detailed strength/weakness strategy matrix developed in the following section of SPEDINFO.

Decision support in this component of the system deals with screening, materials selection, classroom observation and adaptation, and referral. Thus, the system preserves the legal basis for special education programming while being useful for teachers of students of all ability levels.

The information base for this SPEDINFO component is a matrix that relates student characteristics as represented by the strength/weakness profile, to dimensions to instructional strategy as shown in Figure 5. As noted earlier, the links in the above matrix are derived from successful special education practices. The two matrix dimensions are based upon categories {dimensions} used by education professionals in interpreting assessment data and in choosing instructional strategies. Decision support in this component is more sophisticated than in other SPEDINFO assists. The group interactions required for IEP preparation are managed by the support system to promote agreement on critical student needs and the instructional efforts which will be made to meet them.

System Goals - The prototype SPEDINFO system is intended to support the decision-making model depicted in Figure 1. This general goal has four specific decision support goals: support of child placement and monitoring tasks in the classroom; support of primary assessment tasks; support of IEP development; and support of IEP management. Each subsystem addresses one of the decision support goals. Decision support in all of the subsystems will take three basic forms: Decision support algorithms, Decision rules matrices, and Selection lists. Decision support algorithms {DSA} represent a high-level organization of procedural knowledge relevant to the specific goal of the subsystem addressed. They provide an overall context and structure to the necessary decisions to be made and resources available to assist in making those decisions. In general, they take the form of a flow chart which depicts the overall sequence of relevant steps to a solution. Supporting this structure also involves: establishment of levels of detail if the overall logic is complex; provision of guidelines to assist the user in performing procedures; provision of criteria to assist the user in making decisions; presentation of important issues in the field concerning decisions or procedures. The high-level flow chart of this DSA is represented in Figure 1. Decision rules matrices are used when general section procedures are of the form "if condition and if condition then consequent", or when coincident occurrences define a decision rule. The selection of the appropriate instructional materials given a set of educational objectives can be represented by such a matrix, as shown in Figure 6. A selection list is a general term referring to a set of objects and their relevant attributes. These lists occur when it is necessary to select from a set of objects according to some criteria concerning their attributes. A selection list may exist in isolation, in conjunction with another selection list to form a decision rules matrix, or in conjunction with generalized selection procedures. Note that either of these are embedded, in varying degrees of visibility to the user, in the decision support algorithms. The DSA elements are shown in Figure 7.

Child Placement and Monitoring Subsystem - The purpose of this subsystem is to provide support to the classroom teacher for: placing the child in the curriculum; monitoring the functioning of child in the classroom, and making referral decisions. The top level of DSA structure for this subsystem is shown in Figure 8. The next level of detail in the DSA contains subtasks necessary to support those identified in the top level algorithm. These subtasks, along with the necessary supporting data, are shown in Figure 9, 10, and 11. Figure 12 shows the overall hierarchical structure for the subsystem. Given the kernel set of procedures, decision, and necessary supporting data, the relevant decision rules matrices and selection sites can be defined. Important selections sets for the subsystem include: child characteristics, materials charactertistics, strength/weakness profile categories, and achievement levels {+ tests}.

In summary, it is important to keep in mind that this system feature is intended to help or support the professional in making complex and difficult decisions about many children with a variety of problems. The SPEDINFO feature is NOT intended to replace professionals or do their critical decision making for them. The SPEDINFO feature is an adaptive use of computer technology, intended to assist a range of professionals from teachers, administrators, special educators, psychologists, speech pathologists, medical personnel, etc., in dealing with the enormous volume of data collected about children with suspected problems, it can help them make accurate and precise decisions based on the needs of the child and not on what the system can deliver.

A MICROCOMPUTER/VIDEODISC CAI SYSTEM FOR THE MODERATELY MENTALLY RETARDED

Ron Thorkildsen
Exceptional Child Center
Utah State University

Warren K. Bickel
Department of Psychology
Utah State University

Joseph G. Williams
Department of Psychology
Utah State University

Permission to reproduce this article in the NCIE proceedings has been obtained by the editor of Journal for Special Education Technology.

Computer-Assisted Instruction (CAI) in the last decade, has been demonstrated as an effective educational tool. Not only has it been used effectively in teaching normal and bright learners, but recent data has indicated its effectiveness with mentally handicapped learners. Effective demonstrations have included coin summation (Knutson & Prochow, 1970), vocabulary (Nelon, 1972), reading, mathematics, spelling, and verbal skills (Leonard, 1970; Rosenkranz, 1974; St. Aubin, 1976).

The majority of CAI programs for the handicapped have been designed for the mildly mentally retarded who have at least the rudiments of reading and writing skills. A review of the literature has revealed little emphasis on using CAI with moderately mentally retarded or with non-readers. The lack of research in this area is due to the past limitations of computer hardware. To be effective for moderate and severely mentally retarded children, CAI must provide a full range of color, graphic, and auditory capabilities along with a variety of student response mechanisms. Computer auditory capabilities have been very limited, and because of the limited and costly nature of computer software generated graphics, CAI has been implemented with none or very primitive animation. Colby (1973) has stated that CAI allows only a limited number of responses and that we need to "....find a way to rapidly random access both sounds and pictures" (p. 260). Geoffrin and Bergeron (1977) in the conclusion of their report suggested that a larger variety of activities other than animation are needed and that these "....activities must be designed to train generalization reading skills to new words and situations other than computer animation" (p. 12).

Hirschbuhl and Seeman (1975) have reviewed recent developments in CAI and give three recommendations for expanded application of CAI. They state that CAI must provide:

1. The ability to generate, program, and edit interactions with complex real images.
2. A technique for creating and delivering cost-effective graphics and visuals that allow for interaction.
3. The capability to provide overlays on images for sequential question frames. Such questions need to be communicated by audio messages.

Most of the limitations of CAI cited in the literature are related to video capacities. However, the most serious limitation with CAI for the mentally retarded has been the lack of random access audio. Considering that the majority of the moderately handicapped have little or no reading skills, random access audio is important to a CAI system for this population. It is necessary that the CAI system have the capacity to provide instruction verbally. Random access is necessary to provide smooth and fast transitions to remediation material. A recent technological advance that has great potential to allow the implementation and manipulation of audio and video function is the random access videodisc player.

USU Pilot Project

A research project currently being conducted at Utah State University's Exceptional Child Center is utilizing this recently developed videodisc technology.[1] The major goal of the project is to develop and field test a prototype individualized instructional system for moderately mentally retarded learners using the videodisc player interfaced with a microcomputer system. The system developed by this project is referred to as the MCVD (MicroComputer VideoDisc) system. This system will alleviate many of the problems mentioned by Colby (1973) and Geoffrin & Bergeron (1977). The videodisc offers a fast, economical, and feasible way of storing and presenting audio/visual information. Specifically, the videodisc allows the random access of 54,000 frames of audio/visual information in less than eight seconds from any portion of the disc. Accessing segments within a program will require only a fraction of a second. The videodisc allows the use of both still pictures and motion pictures with or without the use of sound.

The major activities of the MCVD project are:

1. The design and implementation at the hardware interfaces between the microcomputer, videodisc, and the touch panel.
2. The modification of PILOT, (an authoring language), to interact with the videodisc and the touch panel.
3. The revision of the "Matching Shapes and Sizes" instructional program to be used with the MCVD system.
4. The filming of the instructional sequences on two-inch video tape to be used in the production of the videodisc.
5. The development of a classroom management plan to facilitate the implementation of the MCVD system.
6. The field testing of the system with moderately mentally retarded non-readers.
7. The analysis and revision of the system as a result of the field test.

MCVD System Components and Operation

The hardware for the MCVD system consists of the MCA videodisc player, the IMSAI 8080 microcomputer, North Star digital disc, Sony 12" color television monitor, and a light-interrupt touch panel. The MCA videodisc was selected because of its random access capabilities. The IMSAI microcomputer and the North Star digital disc were selected because of their modularity and reliability. A color monitor was chosen because of the additional flexibility provided by having color available. The touch screen is a light-interrupt system which allows the learner to interact with the system by touching the screen (see Figure 1 for hardware configuration).

The software for the MCVD consists of the instructional material and the computer programs required to present the instructional material, and monitor the student's response.

The computer language that enables the instructional material to be presented is PILOT. This language is designed specifically for CAI and is useful because educators with little or no computer experience can write CAI courses. PILOT has been modified so that programs written in PILOT can control the videodisc.

The instructional program courseware, developed for use with the MCVD system is based on the instructional package "Matching Sizes, Shapes, and Colors", (Hofmeister, 1977). The matching program is designed to teach the learner to match objects that are alike in size, color, or shape. This Matching program was selected for three reasons. First, the matching task is relevant for learning behavior of greater complexity such as language

---

[1] This project was funded on a pilot basis for one year by the Utah State University Research Office.

(Sherman, 1971); affect (Bijou & Baer, 1961); imitation (Gewirtz, 1971); and reading (Sidman, 1971). Second, the skill of matching to sample is typically taught to moderately and severely retarded learners. And third, this instructional package has been field tested and validated. The modified version of the package was kept close to its original; however, some reprogramming has been required. For example, a puppet and a model have been added to present the instruction and enhance the learners' interest.

In addition to rewording some of the instructions of the original program, feedback responses were developed for correct, marginally correct, and incorrect responses by the learner. The microcomputer categorizes the response, and the puppet gives verbal feedback appropriate to the type of response. The reinforcers consist of movie film clips and filmed sequences involving verbal and auditory feedback and/or praise from the puppet.

The following briefly describes how the system interacts with the learner:

> The system presents an audio instruction and an associated visual image via the television monitor. The learner responds by touching the television screen. The area touched on the screen is transmitted to the microcomputer via a touch panel and the response is evaluated. If the response is correct, a signal is sent to the videodisc player by the microcomputer. This signal references a segment on the videodisc which contains an audio and visual reinforcement (a film clip). Other possible response conditions are a wrong response, a close response, and a non-response. The non-response is detected by allowing a specified period of time to respond. There will be recorded segments on the disc for all possible response conditions as well as a variety of positive reinforcements varying in length and type. There is also remediation segments which can be branched to when appropriate.

Parameters are set in each lesson within a program which determine: (1) the number of times the learner must respond correctly before branching to a reinforcement segment; (2) the number of times the learner is allowed to cycle through the same lesson without reaching criterion before the teacher is signaled and asked to intervene. A time limit is set for each session at which time the teacher is signaled to end the session. Figure 2 provides an example of the instructional sequencing used. The sequence is presented smoothly and without appreciable gaps between sections because of the random access nature of the videodisc player.

A classroom management plan and associated manual will be developed describing the use of MCVD system as part of the classroom environment. This manual will describe how the MCVD system can be incorporated into an existing classroom and how the MCVD system and existing classroom program can complement and supplement one another. This is an important aspect of the project, and one that has not been fully considered in many CAI applications. In many instances critics of computer-assisted instruction have viewed the computer as a solution looking for a problem. Ellis (1974) has stated:

> The computer is but another vehicle to employ in helping people learn....Rarely can one productively start in educational reform without things, whether they are books, films, or computers. One starts with a consideration of the ends of instruction, and employs things only as each of these emerges as the most efficient approach to meet these ends.

The random access videodisc player is a new and powerful media device, but by itself is also one of these things. It has the same potential problem of being a solution looking for a problem. A management plan that emphasizes the attainment of instructional objectives will help insure that the MCVD system is an integral part of the instructional environment and not an imposition to this environment.

Videodisc Production

The production of the actual disc presents one of the major problems in using the MCVD system for computer-assisted instruction. The disc must be pressed in the MCA studios in California. Currently this is an expensive process, costing approximately $1,250 for the pressing of one side of a master disc. This expense and the time involved in pressing the disc (two to four weeks) prohibits frequent revisions to the disc. Once the disc is pressed, the developer must be satisfied with the contents. Because of this, current (R&D) models are impractical in designing and validating educational materials for videodisc applications. Most existing R&D models call for several field tests and consequent revision of the materials being developed. At $1,250 per draft, the cost incurred when using a model requiring a number of drafts quickly becomes prohibitive.

To keep development costs for interactive videodiscs within practical limits, an R&D model is needed which increases the probability that the first or second draft of a package will be effective. Several systems modes have been developed which might be adapted for this R&D model. Fault Tree Analysis (Stephens, 1977) for example, is an approach which uses planning techniques to anticipate and analyze possible errors. The incorporation of this type of an approach into an R&D model would reduce the cost of developing instructional videodisc programs.

It is not within the scope of this project to develop this R&D model, although through our experience we would be able to substantially contribute to its development. In the production of the material for the disc for this project, we attempted to reduce the need for revision by thoroughly rehearsing the program script and trying to anticipate all branching situations and associated feedback.

Other ways of reducing the costs associated with the testing of instructional materials destined for use with the videodiscs would be to present the material during early stages of development on other media systems such as the VIS-I-CON system (Fox, 1979) or the Sony Betamax with a random access attachment. The VIS-I-CON system uses a microprocessor to control the presentation of super 8 film and magnetic audio tape. The Betamax is a videotape player that has random access capabilities. Although both of these systems have slower access time than the videodisc (because of the serial nature of the devices), program material destined for use with the videodisc could be tested using these systems, particularly if the film and audio tape segments were kept short, thus lessening the access time.

Future Considerations

Currently under development at USU's microcomputer lab is a feature which will provide the ability to place material from the videodisc and the computer on the screen simultaneously. This will allow the use of computer-generated graphics and text to be displayed simultaneously with the image and audio from the videodisc. Considering the excellent graphics software currently available, this will greatly enhance the flexibility of the system. Another future consideration is the videodisc's ability to store digital data. With this capability an entire program including the required computer software could be stored on videodisc. The addition of a more powerful microcomputer internal to the videodisc player and a general purpose keyboard would result in a single unit system that would be programmable and have computer graphics capabilities.

REFERENCES

Bijou, S.W., and Baer, D.M. Child Development: A Systematic and Empirical Theory. Vol. 1, New York: Appleton-Century Crofts, 1961.

Colby, K.M. The rationale for computer-based treatment of language difficulties in non-speaking autistic children. Journal of Autism and Childhood Schizophrenia, 1973, 3, 254-260.

Ellis, A.B. The Use and Misuse of Computers in Education. New York: McGraw Hill, 1974.

Fox, R.G. Micro-Processor Based Visual Image Controlled (VIS-I-CON) Instruction Delivery System for Instruction of Deaf and Hearing Impaired. Paper presented at the Association for Educational Communications and Technology, New Orleans, March, 1979.

Fredericks, H.D.B. Data Based Classroom for Preschool Handicapped Children. Report submitted by Teaching Research Infant and Child Center. Monmouth, OR: 1978.

Gewirtz, J.L. Conditional Responding for Observational, Imitative Learning, and Vicarious Reinforcement. In H.W. Reese (Ed.), Advances in Child Development and Behavior. New York: Academic Press, 1971.

Geoffrin, L.D., and Bergeron, P.D. Initial Reading through Computer Animation. 1977. (ERIC Document Reproduction Service No. ED 044 039)

Leonard. J.M. Computer Assisted Instruction for the Handicapped: The Role of the Media in the Education of Emotionally Handicapped Children. 1970. (ERIC Document Reproduction Service No. ED 046 158)

Nelon, E.M. An Evaluation of Computer Assisted Vocabulary Instruction with Mentally Retarded Children. 1972. (ERIC Document Reproduction Service No. ED 090 964)

Rosenkranz, C.I. COMP (Computerized Operational Materials Prescription). 1974. (ERIC Document Reproduction Service No. ED 094 539 174)

Sherman, J.A. Imitation and Language Development. In H.W. Reese (Ed.), Advances in Child Development and Behavior. New York: Academic Press, 1971.

Sidman, M. Reading and Auditory Visual Equivalences. Journal of Speech and Hearing Research. 1971, 14, 5-13.

St. Aubin, R. Evaluation of CAI as Used by Various Handicaps. 1975. (ERIC Document Reproduction Service No. ED 136 775)

Stephens, K.G. A Fault Tree Approach to Management - An Overview. A paper presented to the Military Operations Research Society Conference, Annapolis, Maryland, June 29, 1977.

SYSTEM DESCRIPTION

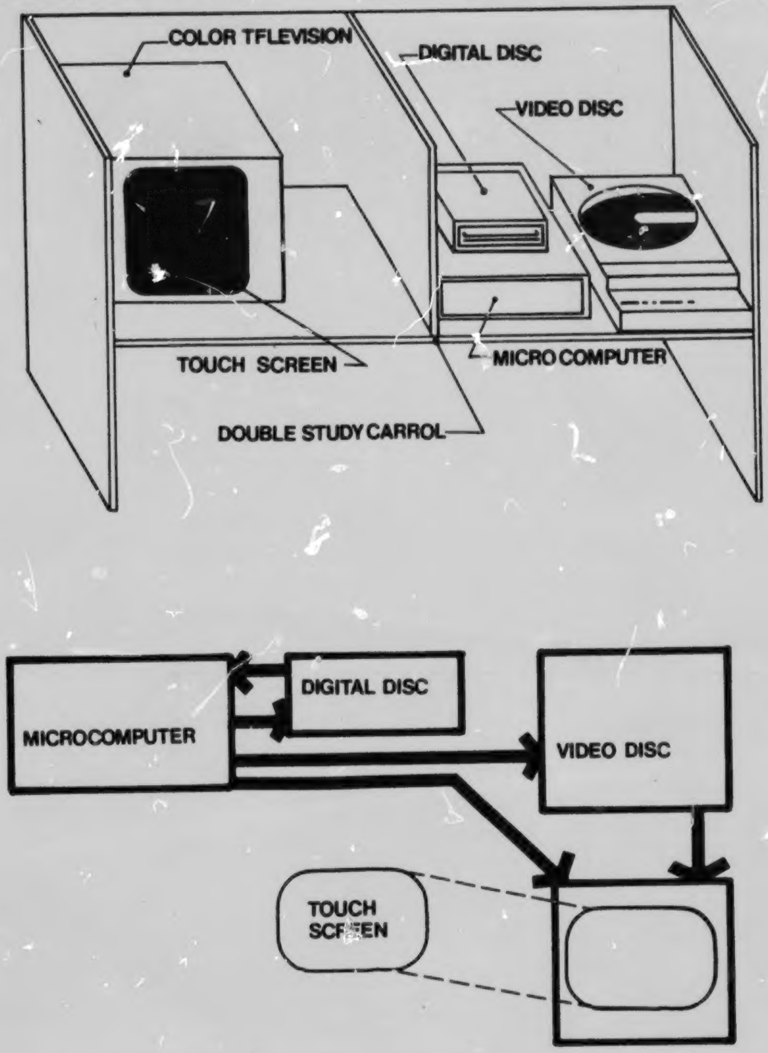

COLOR TFLEVISION
DIGITAL DISC
VIDEO DISC
TOUCH SCREEN
MICRO COMPUTER
DOUBLE STUDY CARROL
MICROCOMPUTER
DIGITAL DISC
VIDEO DISC
TOUCH SCREEN

Figure 1

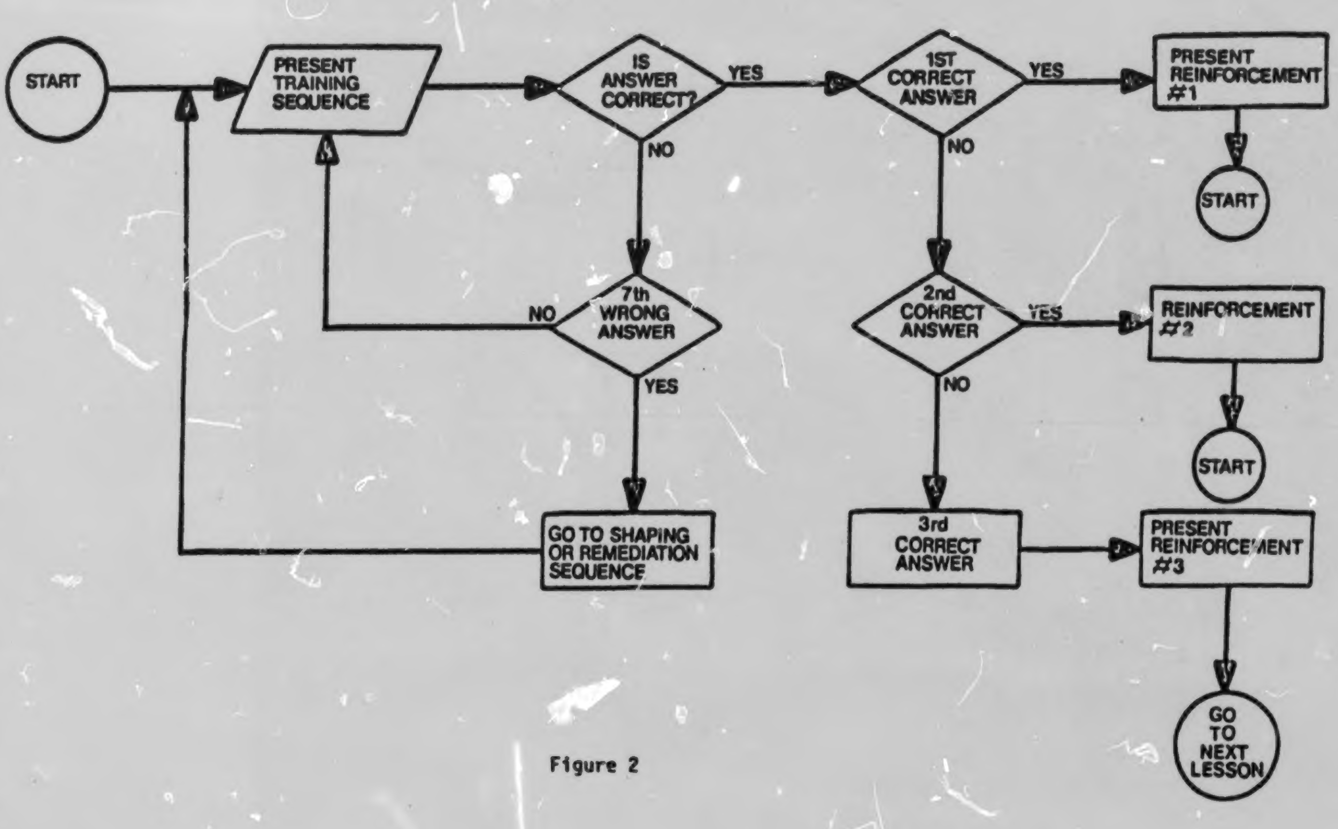

START
PRESENT TRAINING SEQUENCE
IS ANSWER CORRECT?
YES
NO
1ST CORRECT ANSWER
YES
NO
PRESENT REINFORCEMENT #1
START
NO
7th WRONG ANSWER
YES
2nd CORRECT ANSWER
YES
NO
REINFORCEMENT #2
START
GO TO SHAPING OR REMEDIATION SEQUENCE
3rd CORRECT ANSWER
PRESENT REINFORCEMENT #3
GO TO NEXT LESSON

Figure 2

A Computer-Based Laboratory for General Statistics

R. S. Cunningham

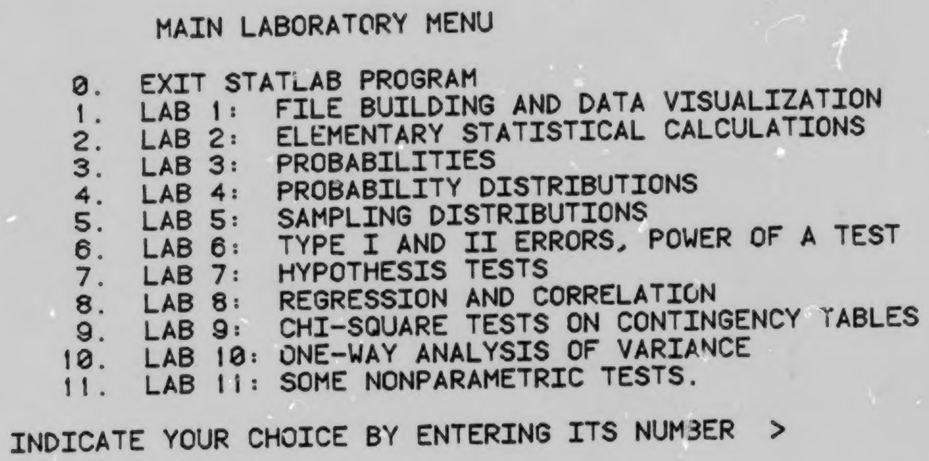
Birmingham-Southern
College

This paper describes a laboratory, including both software and manuals, developed at Birmingham-Southern College for use in our general statistics course. We needed a laboratory for our students that was interactive, needed no computer experience, was oriented toward building intuition for statistical processes, and could be used as a direct supplement paralleling the course. Since none of the available packages met these criteria we chose to develop our own. We believe the resulting laboratory will be of interest to persons in many disciplines who want a computer supplement in their statistics courses.

The laboratory has two functions in teaching statistics. It is a tool which students can use to perform statistical calculations and elementary hypothesis tests, and it illustrates theoretical statistical concepts by generating and reporting examples which verify statistical theory. Both functions, but especially the latter, are supported by computer graphics displays which reinforce numerical concepts and build student intuition. Although terminal or line printer graphics are not uncommon in statistical packages, and frequently are quite good as in the Tanis package[1], ours seems to be the first generally available educational statistics package that uses vector-oriented graphics.

The laboratory was designed to meet several particular goals. It was to be comprehensive, covering all the material in a normal general statistics course as determined by an extensive survey of current texts. The program was to be self-contained, including all needed file procedures including file building. Finally, it was to be friendly and easy for computer novices to use, requiring no computer experience of either student or instructor to be effective. These features with the functions described above generally meet the guidelines for effective packages described by Ling[2], pp. 2-5.

The laboratory is designed to be run parallel to the statistics course. It covers the topics listed in Figure 1, the program's main menu. These individual laboratories include most of the standard



```
           MAIN LABORATORY MENU

    0.   EXIT STATLAB PROGRAM
    1.   LAB 1:  FILE BUILDING AND DATA VISUALIZATION
    2.   LAB 2:  ELEMENTARY STATISTICAL CALCULATIONS
    3.   LAB 3:  PROBABILITIES
    4.   LAB 4:  PROBABILITY DISTRIBUTIONS
    5.   LAB 5:  SAMPLING DISTRIBUTIONS
    6.   LAB 6:  TYPE I AND II ERRORS, POWER OF A TEST
    7.   LAB 7:  HYPOTHESIS TESTS
    8.   LAB 8:  REGRESSION AND CORRELATION
    9.   LAB 9:  CHI-SQUARE TESTS ON CONTINGENCY TABLES
   10.   LAB 10: ONE-WAY ANALYSIS OF VARIANCE
   11.   LAB 11: SOME NONPARAMETRIC TESTS.

INDICATE YOUR CHOICE BY ENTERING ITS NUMBER  >
```

Figure 1

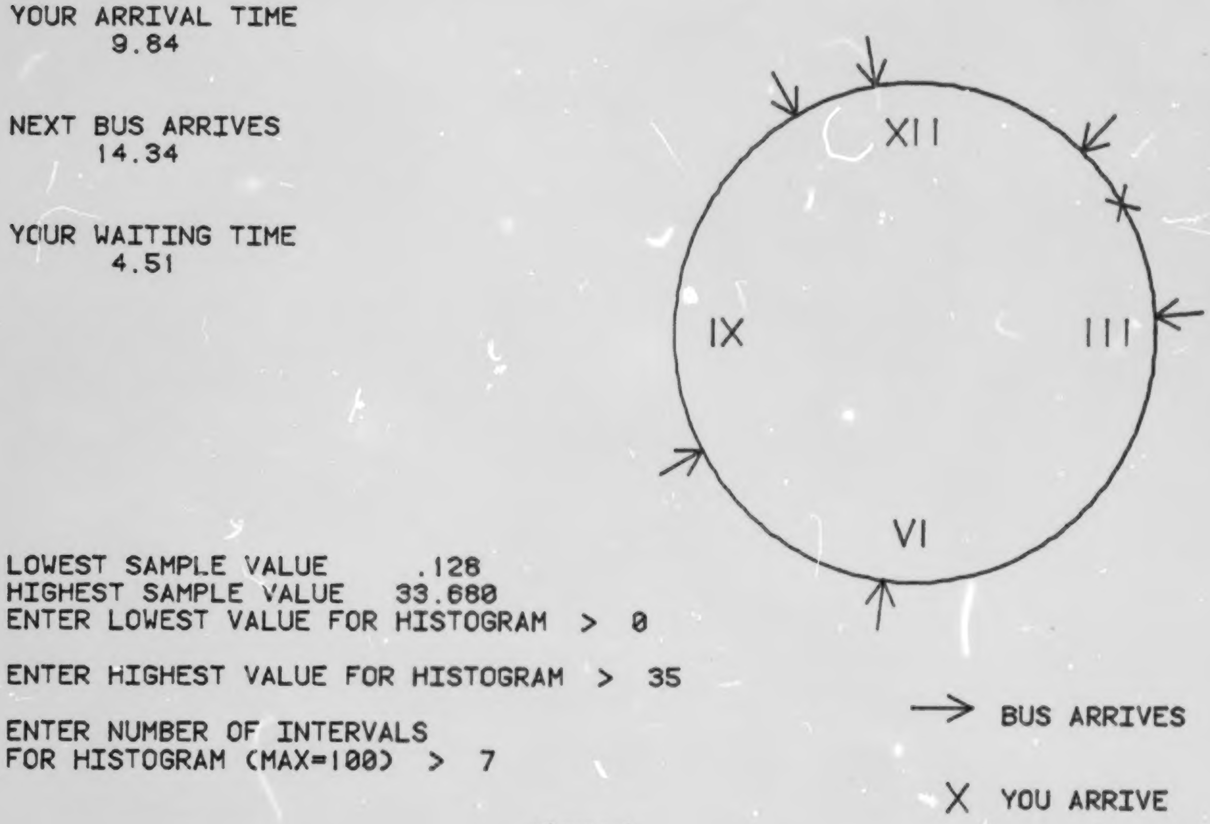
topics in statistics as well as some topics which are irregularly covered, such as Type II errors, power of a test, or nonparametric tests, which can be delegated entirely to the laboratory if needed. One very important aspect of the laboratory, which is in the manual but not the program, is the student data project. This is an integrating experience in statistics as well as a motivation in the course. It has a student pose a question, form a hypothesis, gather data, and use the program to analyze the data and test the hypothesis.

The effect of the laboratory in our statistics course has been quite positive. Students have enjoyed using the computer and have felt that the laboratory helped them learn statistics. We have few objective measurements of its success, but student grades have improved dramatically--by nearly one letter grade on the average--since it was implemented.

As examples of the program's operation, we will describe below a few of the laboratories in more detail, and include some of the graphic displays from these laboratories. These displays are presented in finished form, but in most cases the graph is drawn first and the supporting text follows.

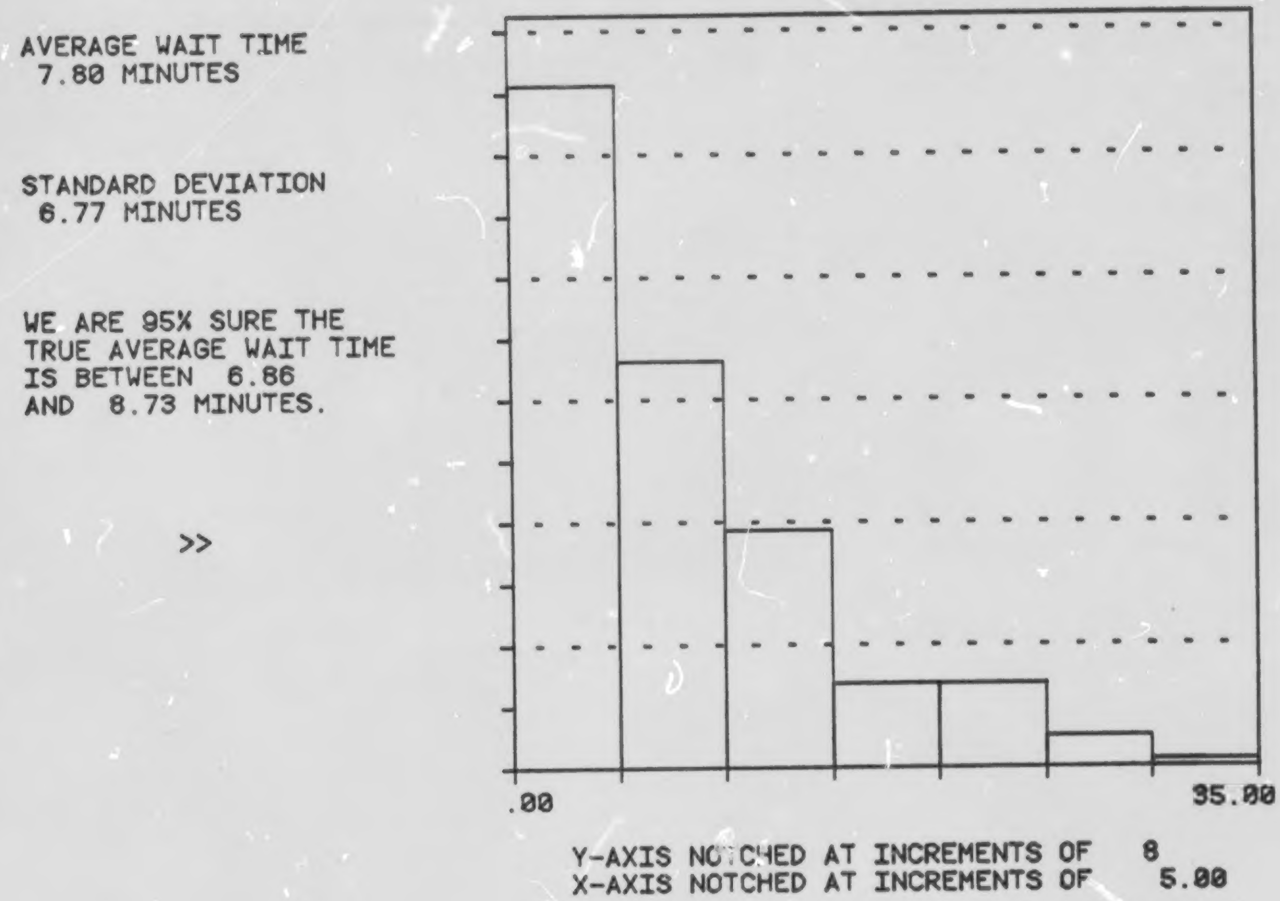
Laboratory 2, on probabilities, approaches the topic from the point of view of simulation. One part of the lab is a bus stop simulation. We assume that six buses arrive at random in a given hour at a bus stop; you arrive there, also at random. What can you say about the time you may wait for the next bus? Theoretically this is a complicated question, and yet the simulation of 200 such events is most revealing. The display in Figure 2 introduces the situation and gives the student information to set up a histogram of the simulated waiting times. The resulting histogram is shown in Figure 3. Notice that the distribution of waiting times is quite clearly exponential, a fact which is not expected by the student. This distribution is then discussed in more detail in the next laboratory.

Probability distributions are studied by sampling. The student chooses a distribution from a menu and is asked to supply the parameters needed to define a particular population with that distribution. One hundred samples are then drawn from the population and the student constructs a histogram of the sample. The histogram, however, is graphed with the population distribution superimposed so the student



```
YOUR ARRIVAL TIME
      9.84

NEXT BUS ARRIVES
     14.34

YOUR WAITING TIME
      4.51

LOWEST SAMPLE VALUE       .128
HIGHEST SAMPLE VALUE    33.680
ENTER LOWEST VALUE FOR HISTOGRAM  >  0

ENTER HIGHEST VALUE FOR HISTOGRAM  >  35

ENTER NUMBER OF INTERVALS
FOR HISTOGRAM (MAX=100)  >  7
```

Figure 2



```
AVERAGE WAIT TIME
 7.80 MINUTES

STANDARD DEVIATION
 6.77 MINUTES

WE ARE 95% SURE THE
TRUE AVERAGE WAIT TIME
IS BETWEEN  6.86
AND  8.73 MINUTES.

      >>

Y-AXIS NOTCHED AT INCREMENTS OF    8
X-AXIS NOTCHED AT INCREMENTS OF    5.00
```

Figure 3

may see how the sample reflects the population. An example is given by the normal distribution sample shown in Figure 4.

Discussions of Type I and Type II errors are important in the understanding of hypothesis testing but are frequently skimmed or even omitted in introductory statistics courses. Our laboratory on these topics allows the student to define a normal test for a mean and examine these errors for that test. The student gives the hypothesized and alternate means, the test's significance level and sample size, and the sample standard deviation. The graphic display resulting is shown in Figure 5. The areas associated with Type I and Type II error probabilities are indicated by solid and dashed shading of their respective curves and the probabilities are printed below the display.

Our final example is in hypothesis testing. The display in Figure 6 reports the results of a test for equal means, but similar displays are given for all hypothesis tests on means, medians, and proportions. The user defines the type of test and provides the test statistics; the resulting display gives a graphic illustration of the test results together with descriptive and literal information giving a summary of the test.

The laboratory's program is quite portable and is written in standard FORTRAN except for a few machine-dependent subroutines defining the program environment and dealing with files. The graphics used are from the Tektronix PLOT-10 Terminal Control System operating at the DRAW/MOVE level and can readily be converted to other graphics systems. We plan to adapt the laboratory to Tektronix' new Interactive Graphics Library system as well when we complete this installation. The laboratory also uses portions of Eliot Tanis' statistics package[1] for basic statistical calculations. We have submitted the laboratory to CONDUIT for evaluation and possible distribution.

```
100 SAMPLES

NORMAL DIST.
MEAN IS    10.00
ST. DEV. IS     5.00




>>
```

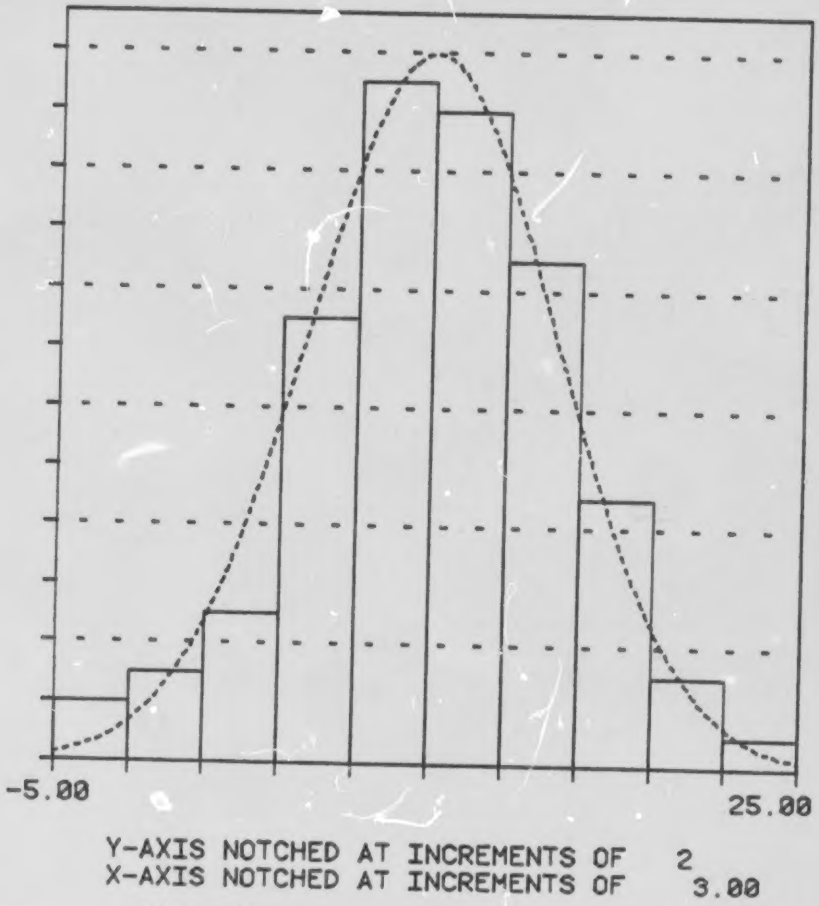
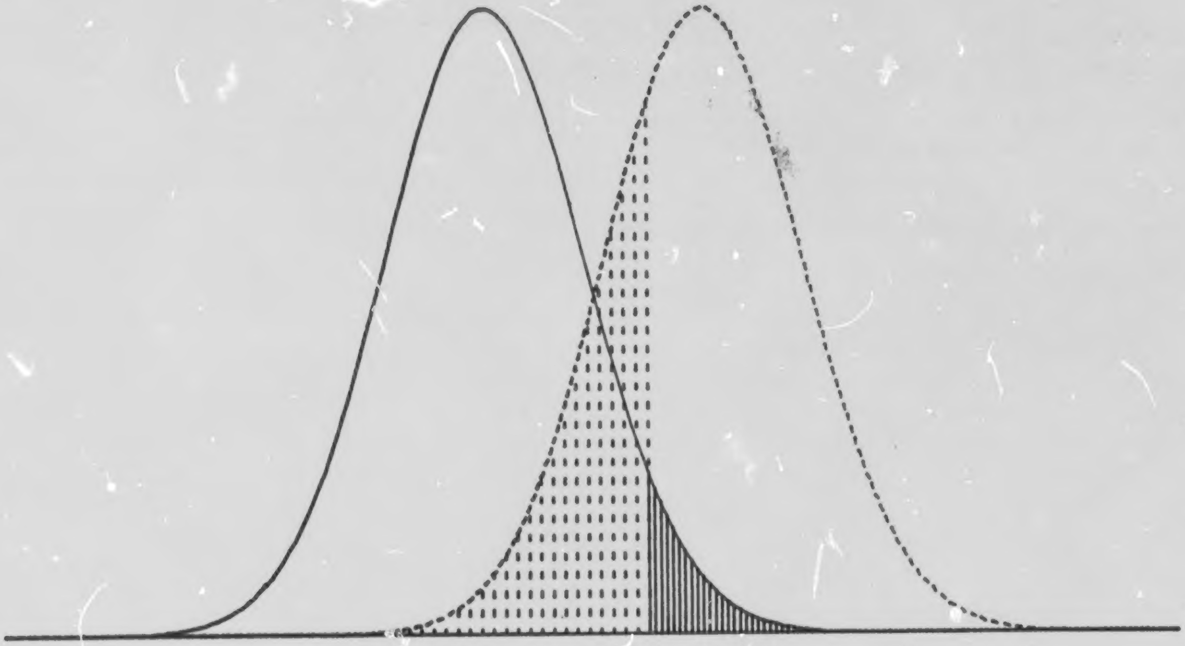


Figure 4

References

[1] Tanis, Eliot A. Laboratory Manual for Probability and Statistical Inference. Register No. STA036, CONDUIT, 1976.

[2] Ling, Robert F. "Guidlines to the Use of Computers in Statistical Instruction". Clemson University Technical Report #288, Sept. 1978.



```
HYPOTHESIZED MEAN  50.000              ALTERNATE MEAN  60.000

RIGHT TAIL TEST

PROBABILITY OF TYPE I ERROR   .050
PROBABILITY OF TYPE II ERROR .282              >>
```

Figure 5

```
HYPOTHESIS:
EQUALITY OF MEANS.
TEST:
TWO TAIL.
SIGNIFICANCE LEVEL:
 .0500
TYPE OF TEST:
NON-EQUIVARIANT SAMPLES
NORMAL TEST.
CRITICAL REGION(S):
Z >  1.9604 OR Z < -1.9604
SAMPLE STATISTIC:
-1.255
PROBABILITY OF A VALUE
SMALLER THAN YOUR
SAMPLE STATISTIC IS:
 .10473


 >>
```

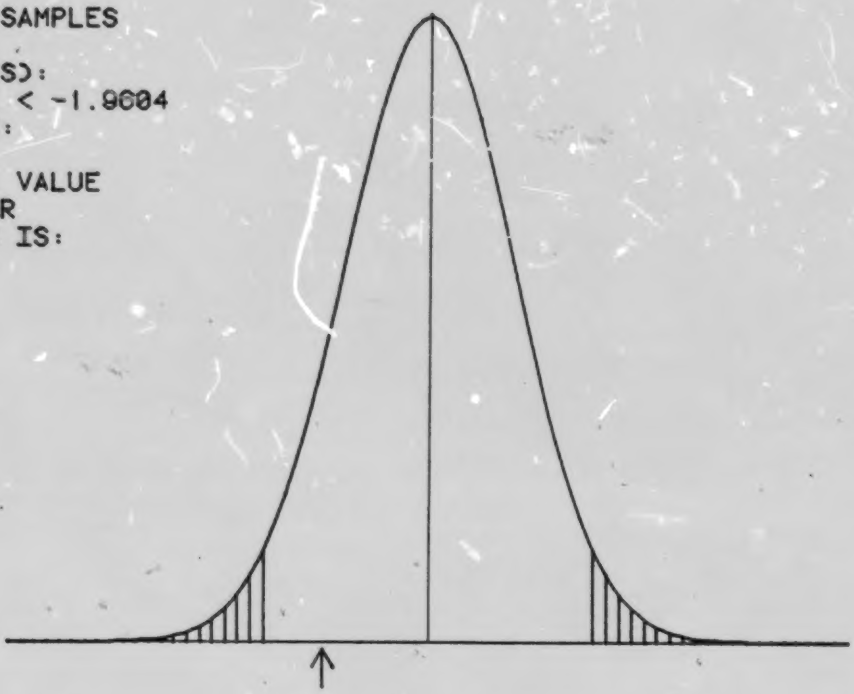


Figure 6

TEACHING STATISTICS AND USING COMPUTERS

Robert Oberg and
Kenneth Preskenis[1]
Department of Mathematics
Framingham State College
Framingham, Mass. 01701

Each college or university offers many statistics courses but there has been limited satisfaction so far with the introductory statistics courses that have been designed and presented.[1] A major factor is that many of these students have attitude problems about mathematics; in fact, many have built-in fears of mathematics. It seems important for these students to have contact with a teacher who tries to get at these problems and can help break down constraints to learning statistics which are due to mathematical inadequacies. A second factor is the nature of statistics; it is not entirely a mathematical science. For instance, a report supporting research in the mathematical sciences puts statistics in a very special category observing that "statistics has a dual source of identity and intellectual force, only one of which is mathematical...statistics is both a mathematical science and something else."[2] Additionally, the report suggests the something else is tied to "the challenge of inference in the face of uncertainty and to the stimulus of the quantitative aspects of the scientific method".

This work offers two approaches to teaching elementary statistics which have proved successful on the undergraduate level at Boston College and Framingham State College. Each approach uses a computer (or a computer terminal) and each addresses the two factors mentioned above as obstacles to learning statistics. Speaking from the teacher's point of view each approach can be described as a process to convey the substance of statistics to students with attitude problems about mathematics.

I. THE ORTHODOX APPROACH WITH AN INTRODUCTION TO THE COMPUTER

The course described here was presented at Boston College in 1976-77 to classes having between 20 and 40 students. The computer support consisted of a fine student-oriented computer center containing an IBM 370 computer with timesharing capability for a large number of terminals. Based on the author's experience over several years in a small college, however, the course could have been given at a college that had one terminal linked (over a telephone line) to a smaller computer. This is because actual computer presence in the instruction occurred through a handful of relatively unsophisticated computer programs which the student simply executed. The programs were well prepared and were written in BASIC. The student did not have to write any programs and was not required to know a programming language. In fact, after finding the computer center and a terminal in it, the student simply pressed a couple of buttons and/or switches, typed an identification number, and the word "run". The main point was that each computer use was an event carefully integrated into a larger organized instructional unit on statistics. Thus, this approach gave due respect to the fact that the computer as a gadget introduces a source of distraction for the student that might take the student's attention away from the substance of statistics. On the teacher's part, although there was no need to get into technology, there was a non-trivial adjustment in allowing the computer to contribute to a student's insight.

A major concern in the design and presentation of this course was with the role and the image of the student. For instance, it seemed almost futile to try to introduce computers and statistics to students who have never had a meaningful mathematical experience and who do not want one unless the teacher can also remove some of the obstacles blocking the way. The mathematical part of statistics requires abstract reasoning and the teacher should make an effort to help the student do that mental activity. Thus, the aim of this approach was to produce an instructional process in which the role of student is more attractive and which offers some compensation for students who have weak backgrounds in abstract reasoning. The textbook Elementary Statistics by Hoel was specified and used, but the teacher also constructed handouts explaining the main points in each chapter. This was significant. The handouts minimized note taking in class sessions, offered readily available examples of a continuous line of thought in statistics, got the teacher more involved with the material, and finally, demonstrated that the teacher was trying to go to where the students are. Additionally, one week before each hour exam a sample test was given, and this sample was used to review the material just before the actual exam. Finally thanks to a collection of equivalent problem sets put together by Professor Paul Banks, each student could make up any homework assignment with an alternate assignment, but only for 75% of its original value. Thus, enough supporting structure was supplied to insure success for any student giving an honest effort.

The first part of the course covered the data gathering and descriptive techniques of statistics as well as some elementary probability. The first assignment required each student to have fifteen simple questionaires filled out by a sample of their choosing. The students' results led to class discussions about the samples and to conjectures about the Boston College population as a whole, e.g., what is the proportion of males at Boston College in view of the specific results in the sample. Numbers and variables came out of the questionaire including some variables used as devices for quantifying non-numeric objects such as human qualities or attitudes. This led to a comparison with the more mathematical view of numbers considered as first degree abstractions (the size of sets) and of variables considered as second degree abstractions (thinking about thought). Drawing the distinction between the degrees of abstraction took a couple of class sessions and was integrated with a brief review of selected high school material. This helped the student more fully appreciate the concept of variable. Next, grouping values of a variable x arose in a natural way from the questionaire; this led to the frequency variable f and the relationship between the two variables expressed in graphical and table form. At this stage a computer program supplying the exact mean and standard deviation was used together with a careful comparison to the hand calculated approximations which came from the grouped data and the so-called class marks.

Probability models of coin and dice tossing experiments were presented with emphasis on the tie between equally likely physical outcomes and the psychological notion of equally likely. Class sessions cultivated the idea that the theoretical probabilities agreed with intuitive projections of those probabilities. There was a different aspect of the computer used in this context, namely, the random number generator, which yielded a computer simulation program for randomly drawing cards from a container. With this program the student could calculate the empirical probability of an event and compare it to the theoretical probability of the same event in the probability model.

The topics of binomial distribution, normal distribution, hypothesis testing, confidence intervals, and point estimates were treated using many problems and examples to illustrate the ideas. A full week was spent on the approximation to the binomial distribution by the normal distribution emphasizing the mathematical idea of replacing discrete information by a smooth curve. Finally, it was decided to present enough ideas from correlation and regression to get more mileage from the student questionnaires and also to construct the linear forecasting model that ties the logarithm of the population of the U.S.A. to the years from 1800 to 1950. One simple number-crunching program was used to supply the correlation coefficient and the slope of the regression line equation in response to student-submitted sample data.

A specific benefit of the computer uses was that the students acquired a deeper understanding of the point of view of applied mathematics. In particular, there were problems which seemed to be outside of mathematics — such as how is a person's political attitude related to the attitude about the legalization of marijuana — which, after some working hypothesis, could be cast into a linear regression model. And wherever the results of the model agreed with our experience, the model could be used as a working model. The computer contributed to the construction of the model by supplying calculations quickly and by allowing a crude model to be refined (or if necessary, scrapped in favor of an alternate approach).

Looking back on the course, an error of omission was not to have treated the algorithm and accompanying flow chart for any of the programs. This would have been beneficial to the teaching-learning process. Nevertheless, a valuable by-product of the course was the implanting in the student of some feeling for how the computer is used and what some of its limitations are.

The only factual information for evaluating the course is the collection of course evaluation questionaires which are filled out voluntarily by the students. 80% of these questionaires rated the course high with not one negative comment about the computer uses. The author's interpretation of this response is that the students benefited from the increased attention given to their individual problems with learning statistics, and that the computer uses were successfully integrated into the course. Some common remarks from different questionaires seem appropriate as statements of what some students found valuable:

The course had a casual atmosphere-it was not life or death.

The grading criteria were very fair. The homework and test were good learning experiences.

Many real life examples were used in the teaching process.

The instructor was willing and able to adopt his teaching and the course to the strengths and weaknesses of the students.

II. THE EMPIRICAL APPROACH

The course described in this part of the paper was taught at Framingham State College to classes of 50 students and of over 100 students during the period 1975-78.

1. The STATLAB Course

Hodges, Krech, and Crutchfield have developed an interesting approach to teaching elementary statistics called STATLAB.[3] The heart of the concept is a real data base, called the STATLAB Census, that is reproduced in the students' textbook. The Census consists of 32 variables—physiological, psychological, and socioeconomic—recorded for 1296 families. Students work with these real data. The whole Census is treated as the population, and each student works with an individual random sample of size 40 (obtained by throwing dice). Since each student's sample is unique and different from his or her classmates', the inherent variability of statistics is well brought out.

As homework students make various calculations with their own random samples, for example finding means (and medians). Each student arrives at a different value of $\bar{x}$, and by pooling these values a class distribution of the $\bar{x}$ statistic can be obtained. A histogram for $\bar{x}$ is then constructed and the standard deviation of $\bar{x}$ is calculated and compared with the standard deviation of the original x. This leads to a more effective demonstration of the central limit theorem than is possible in the conventional course.

As a second example of this empirical learning process, each student calculates a 95% confidence interval for his $\bar{x}$, and then the instructor reveals the true value of the population mean. (The proper timing of this revelation is important.) By a show of hands it can be seen how many of the class have captured the true population mean in their confidence intervals, and it (hopefully) turns out to be close to 95%. By working with individual samples and comparing results in class discussions, students develop good insight into many of the important concepts of statistics.

2. An Extension of the STATLAB Approach Using Computers

The author has used STATLAB four times in teaching elementary statistics and is convinced of its positive effects on the students' understanding of concepts; futhermore, students seem more interested in working with real data than in following a more traditional textbook approach.

There are some practical problems. Since each student has his or her own unique sample, how can he or she know if computations have been done correctly. It is infeasible for the instructor to check each student's calculations by hand, so what is to be done? The lack of feedback on the accuracy of their work can quickly diminish student interest, and it may happen that in later assignments only a minority of students will have performed the calculations at all, making class discussion impossible. An underlying assumption of the STATLAB concept is that most of the students keep up with the work, making possible comparisions and pooling of results. When this assumption breaks down the instructor can become discouraged. This discouragement may account for the STATLAB approach being abandoned at some institutions after initial enthusiasm.

During the first semester of using STATLAB the author became painfully aware of the above problems. The computer seemed like a logical candidate to offer a solution. A system of computer programs called STATCLASS was developed that performs the following functions:

(1) Individually check each student's calculation.

(2) Assign grades for each assignment and keep track of all grades in a computer file.

(3) Construct frequency tabulations of statistics like $\bar{x}$ (each student contributing one value based on his or her own sample).

The first step in setting up the computer system was to get the Census data base into the computer. Through correspondence with one of the authors of STATLAB[4] it was ascertained that the Census data base was not immediately available on cards or magnetic tape. The fastest way to get the data up locally on the computer was to have students keypunch the data. Each student was instructed to keypunch, in a prescribed format, the data for his or her own sample of 40. (Actually, only 17 of the 32 variables in the Census were used in the text, and only these were keypunched.) A program was written to read these decks of 40 cards, check for legal format, and write the information to a file. When a particular record was already in the file, a check was made that the information already recorded agreed with the information being submitted. In this way a fairly accurate computer file of the Census data base was built up. The file, henceforth called CENSUS, was completed and checked in the second version of the course. One lesson learned by having the students do all the keypunching for the data entry is that students can be quite cooperative with the teacher in such a project (especially when given a small amount of course credit for it), making it quite feasible and inexpensive to build up locally a computer data base.

3. The STATCLASS Computer Programs

The programs are written in FORTRAN and run on a CDC Cyber 72 computer using the NOS Operating System. The largest programs require approximately 65K words of central memory. The programs make use of three sequential files. The first, CENSUS, was described above. It consists of 1296 records; each record is identified by a 4-digit family ID number and contains values of 17 variables. Second is CUMFILE, the cumulative grade file. It has one record for each student in the class, containing student number, name, major, class, and homework grades. Third is SAMFILE, the sample file, which also has one record for each student, containing student number, name, and 40 family ID numbers (identifying to the computer the student's sample of 40 that he or she drew from the Census population).

There are five principal programs in the STATCLASS system. They are all run by the instructor using as input cards punched as assignments by students in the class. There are also several utility programs used for tasks such as sorting input decks.

SETCUM. As the first punched card assignment, each student is asked to keypunch one card with his or her student number, name, major, and class. SETCUM reads a deck of these cards and initializes CUMFILE. Five points of homework credit are given each student for submitting a properly punched card.

SETSAM. As the second punched card assignment, each student is asked to keypunch three cards giving student number, name, and the 40 family ID numbers identifying his random sample of 40 families. SETSAM reads a deck of these cards and sets up SAMFILE. Also the CENSUS file is read and a report printed for each student showing the data recorded in the computer for selected variables for each family in the student's sample. This allows the student to check that the data have been properly recorded from the STATLAB Census. This is important because when later (various) calculations are checked, the values the computer will use for a given variable are those recorded in the CENSUS file. The student must be sure to work with these same values. Finally SETSAM creates a temporary transaction file for input to the UPDATE program described below. The transaction file contains one record for each student doing the assignment. Each record contains a student number, assignment number (two in this case), and score (ten points are awarded for this assignment).

UPDATE. This program is run after the main program for each assignment beginning with the second assignment. It reads CUMFILE and the temporary transaction file. By matching student numbers from the two files, an updated CUMFILE is created reflecting the points earned from the current assignment. This is the program that keeps track of homework grades; it is very useful when teaching a large class.

MEANMED. For their third assignment students keypunch the values they obtain in calculating the mean and median for two selected variables MEANMED prints out for each student a report showing the correct values of each statistic for his sample and the intermediate work leading up to the correct answers, including frequency tabulations. Also a grade is assigned based on how close the student's reported answers are to the correct answers. These grades are printed as part of the student's report and also are written to the temporary transaction file for use by the UPDATE program in updating CUMFILE. In order to prepare the reports MEANMED has to read SAMFILE to get the 40 family ID numbers of the student's random sample and CENSUS to get the proper values of the variables in the student's sample. MEANMED also writes the correct values of the calculated statistics to another temporary file, the answer file. An SPSS program[5] is used to read the answer file to produce frequency tables of the $\bar{X}$ and $\tilde{X}$ statistics for use in class discussion.

SDSE. For their fourth assignment, students keypunch the values they obtain in calculating the standard deviation and standard error of the mean for the same two variables used in the third assignment. SDSE does the same type of processing with the fourth assignment deck as MEANMED does with the third assignment deck.

4. *Discussion and Evaluation*

The substance of the introductory statistics course using the STATLAB approach augmented by the STATLAB programs is quite close to the traditional course, but the process of learning is significantly altered. By working with their own random samples from a real population, students appear to gain a better understanding of key statistical concepts. By giving the students computer-assisted feedback on the accuracy of their individual calculations, needed student involvement with the continuing coursework is maintained. Based on conversations with many students, the empirical approach had a positive effect on attitude, mitigating the anxiety many students felt about taking a required course from the Mathematics Department. Results are available from class evaluations conducted the last two times the course was taught. Responses to the three general questions are shown in Table 1. Ratings are on a scale from 1 to 5 with 1 = least favorable and 5 = most favorable.

Table 1

Median Ratings in Class Evaluation	1976 (n=89)	1978 (n=32)
1. How much do you feel you learned?	3.81	3.94
2. How interesting did you find the course?	3.41	4.03
3. How did you like the STATLAB approach (working with realistic data) compared to orthodox textbook?	4.39	4.80

It should be pointed out that the STATCLASS programs represent only a first step in computerizing the STATLAB approach. Presently only a portion of the students' calculations are checked, and students interact only indirectly with the computer through turning in punched cards to the instructor. It is anticipated that the STATCLASS programs will be extended by widening the range of calculations performed and opening up the programs to direct student access.

Footnotes

1 *Introductory Statistics Without Calculus*, a booklet by the Committee on the Undergraduate Program in Mathematics of the Mathematical Association of America.

2 *The Mathematical Sciences: A Report*, by the Committee on Support of Research in the Mathematical Sciences on the National Research Council for the Committee on Science and Public Policy of the National Academy of Science, Washington, D.C., Publication 1681 (1968), pp. 84 and 157.

3 *STATLAB: An Empirical Introduction to Statistics*, by J.L. Hodges Jr., David Krech and Richard S. Cruthchfield (published by McGraw-Hill, 1975).

4 A Personal Communication with J.L. Hodges Jr., September, 1975.

5 *Statistical Package for the Social Sciences or SPSS*, by Norman H. Nie et al (2nd edition published by McGraw-Hill, 1975).

Note

All correspondence and requests for program data should be directed to Professor Preskenis.

A PROBLEM-SOLVING MODEL IN CONTINUING EDUCATION FOR DENTAL PRACTITIONERS

Laurette F. Lipson, M.N.Ed.
Kenneth D. Keith, Ph.D.
Gary L. Jones, D.D.S.

University of Nebraska

INTRODUCTION. Dentists throughout the state of Nebraska and senior students at the College of Dentistry at the University of Nebraska have a unique opportunity to experience problem-solving and decision-making activities through the use of dental computer simulations on the PLATO system. This opportunity derives from a three-year grant awarded to the College of Dentistry by the National Library of Medicine, and is being developed in conjunction with the University library. The goal of the project is to study the effectiveness of computer simulations and supportive library materials as a resource for improving decision-making and problem-solving skills in dental practitioners.

Certain emergency dental situations offer the best example of the need for comprehensive problem-solving skills. These situations demand basic knowledge and the ability to acquire and interpret information leading to a diagnosis of the patient's problem. The dentist must also select the proper treatment and evaluate the progress of the therapy. There are, of course, times when the initial decision is incorrect requiring a cyclic approach to the problem-solving process. Although this process occurs regularly in private practice, the growth of these skills is often limited in the academic setting. Here, a single student is usually able to observe a particular clinical case. The quality of this experience may be inhibited since the clinical instructor typically makes the major judgments and decisions. Accordingly, only one person in a class may have experienced the unique case, and in reality may not have made any of the major judgments or decisions. For this reason, it becomes essential to find ways of continuing to develop professional judgment even after graduation. Simulations offer realistic situations in which to make decisions with minimal risk. Basic information and classic situations, as described above, can be standardized while allowing a range of individual responses and choices.

NEBRASKA DENTAL SIMULATIONS. At Nebraska such computer-aided dental simulations are supported by PLATO V terminals with touch-responsive screens and random access slide projectors for selection of radiographs, oral and extraoral photographs, functional study models, and patient health histories. The touch capability of the screen allows special features such as: palpation of oral structures, selection of teeth for dental tests, extractions, and keyboard-free decision-making. Each simulation begins with a statement of the patient's problem as it might be reported via telephone. The user can see an actual picture of the patient as s/he is seated in the dental chair and must decide where to begin in the data gathering process. The participant may elect to see further intraoral or extraoral views of the patient, take a medical or dental history, view radiographs, study models, or perform various tests (e.g., vitality, percussion, or hypersensitivity) on the teeth. When the diagnosis has been developed, the dentist may elect to treat the patient, consult with or refer the patient to a specialist. A wide range of possible choices are included in the simulation, allowing the dentist the opportunity to make both correct and incorrect choices. Feedback to the user occurs in a natural way, as the response of the patient to therapy is simulated.

Following each simulation, a summary describes the treatment actually employed or suggested by the author, the key to the proper diagnosis, and library references germane to the problem. These references may address the epidemiology, etiology, diagnosis, or treatment of the disease or condition presented in the simulation. Each reference includes library call numbers and a one sentence abstract of the article so that the user may readily select those in which s/he is most interested.

PARTICIPATION OF PRACTICING DENTISTS. The pool of participating dentists is being drawn through demonstrations at state and local meetings, individual letters, presentations to small groups, and word of mouth by early participants. Since the facilities are being used for development at the same time, evening sessions have been scheduled for local dentists and Saturday sessions are scheduled for dentists from other parts of Nebraska. Individual dentists and college faculty are accommodated as their schedules permit.

Professional Profile. Each participant is asked to fill out a professional profile which includes demographic information about the individual, a self-appraisal rating scale assessing interest, desire for improvement, confidence in data gathering abilities, and confidence in treatment abilities in nine areas of dentistry. Finally, the dentist is asked to complete a log of four sample days of his/her practice, using the nine areas of dentistry to identify each patient's diagnosis and indicating if treatment was performed or the patient was referred to a specialist. The occurrence of diagnoses and referrals of specific therapies are weighted from the most frequent to the least frequent.

Individual Profile. This information is translated onto a grid so that the weights for each area can be totaled. This grid represents the participant's individual profile, and the three highest scores become the areas in which simulations are assigned (Figure 1). In this example, the dentist would be assigned problems in periodontics, orthodontics, and oral surgery, in that order.

	OCCLUSION	ENDODONTICS	FIXED PROSTHODONTICS	OPERATIVE DENTISTRY	ORAL SURGERY	ORTHODONTICS	PEDODONTICS	PERIODONTICS	REMOVABLE PROSTHODONTICS
Occurrence Ranking	3.5	1.5	4.5	5.0	4.0	2.5	1.0	5.0	2.0
Referral Ranking		5.0				4.5		4.0	
Improvement	2	2	2	2	3	4	3	4	2
Interest	4	3	4	2	3	2	2	3	4
Data Gathering Abilities	1	2	2	2	3	4	4	4	2
Treatment Abilities	1	2	2	1	3	4	4	4	2
TOTAL	11.5	15.5	14.5	12	16	21	14	24	12

Figure 1. Sample Individual Profile

Classification of Simulations. Although the number of simulations is limited at this time (18) their scope is broad and each problem is classified in several categories and can be assigned in more than one area (Figure 2).

	Occ	Endo	Fixed	O.D.	Oral Surg	Ortho	Pedo	Perio	Remov	Systemic
U - C Neb 1	S	S			M					S
T - C Neb 2		S			S			M		
T - C Neb 3		M						S		
T - C Neb 4	M	S								
U - A Neb 5		S			M		S			
U - A Neb 6		S			S		M Diag			
U - A Neb 7					S					M Diag
U - A Neb 8	S	S			S			S		M Diag

Figure 2. Classification of Simulations

M - indicates the major area of treatment for this problem.
S - indicates areas which should be ruled out or which mimic the real diagnosis.
T - indicates a problem which may typically be seen in a general practice.
U - indicates a problem which is unusual and not generally seen in everyday practice. These problems are especially useful for differential diagnosis, sustaining interest in the learning experience, and extending knowledge about unusual conditions.
A - indicates a problem of an acute nature.
C - indicates a chronic type of condition even though the presenting symptoms appear acute.

Assigning Simulations. A few guidelines have been established for making the individual assignment:

1. Assign 2-3 simulations in each area for a total of seven. Expect the participant to complete five, and have two additional ones available if needed and if the participant is interested in continuing.
2. If a clear-cut area is evident (one score noticeably higher), the highest number becomes the first area of assignment, the second highest becomes the second area, and the third highest becomes the third area. Provide one or two unusual problems to maintain interest.
3. If scores show confidence of present handling but still express interest and desire for improvement, I assign unusual problems in those areas.
4. For students no rankings for occurrence and referral will exist. Assign areas with the highest numbers. Provide one or two unusual problems to extend knowledge and maintain interest.
5. If data gathering and treatment scores reflect low self rating, provide both typical and unusual problems in those areas.
6. If no clear area can be identified, or if there is confusion in assignment for any particular individual, determine a mutually agreed upon assignment with the participant.

The assignment is generally made by one Project investigator. An independent reliability check by the evaluator shows an 80% overall reliability in the selection of simulations.

Each participant is provided with a series of 5-7 simulations which match his or her desire for change, interest in continuing education, and need for improvement in data gathering and treatment abilities, balanced against the reality of the occurrence and referrals in those areas. This personalized assessment and learning plan is based on the medical model developed by the Illinois Council on Continuing Medical Education for use by physicians.[1]

Participants. Thirty-six dentists from various locations in the state have participated so far. Every participant is given an individual orientation to the PLATO system and a practice simulation before beginning his or her assignment of problems. Following the orientation, each dentist has completed approximately six hours of experience at the terminal, and a minimum of seven simulations. None of these dentists have had any previous experience using computers or any form of computer-assisted instruction. Their ages range from 25 to over 50, with the median age being 42, and with years of practice experience ranging from one to over thirty years. The majority of these dentists have been enthusiastic enough to request additional participation when more simulations are completed and available.

EVALUATION. Evaluation of the Nebraska simulations focuses on the decision-making behavior of participating dentists as they solve the clinical problems presented. Steps in the evaluative process include the determination of optimal and acceptable solutions, assignment of a score to each possible decision in each simulation, recording and plotting of individual pathways, and comparison of individual pathways to preferred pathways.

Scoring: Criteria. Assignment of scores is fundamentally dependent upon the author's judgment in defining acceptable, unacceptable, and optimal decisions within the simulation. This judgment is aided by three experienced dental consultants who serve as a review panel and who critique each problem prior to its use by participants. Three such review panels have been convened so far. When the problem has been reviewed and acceptable pathways identified, specific scores are assigned to each decision in the simulation.

Basically, the import of each decision lies on a continuum extending from absolutely essential (i.e., a good decision of critical importance) to life-threatening (i.e., a bad decision that could be fatal). These extremes were used as starting points from which to complete the continuum, with corresponding assignments of scores. The scoring criteria were developed using guidelines presented by McGuire, Solomon, and Bashook[2] and are summarized in Figure 3. Project staff assist each author in reviewing all possible decisions, applying the scoring criteria, and assigning a score to each decision in the simulation. The result is a scoring guide for each clinical simulation.

Time and Cost. Many clinical decisions result in the expenditure of time and money by patients. Recognizing this fact, time and cost data have been built into the Nebraska simulations. Thus, when a decision is made to see a patient, to carry out a simulated dental procedure, to refer or treat, the time and cost to the patient are recorded. The values recorded are calculated to reflect fees and times prevailing in the local community, allowing computation of total simulated expense and clinic time for each patient simulation.

Cumulative Records. Participant data from the simulations are plotted as cumulative point totals (scores) against dollars or time. Figure 4 is an example of a participant's cumulative record for a single simulation. The cumulative record is useful as a tool capable of presenting a continuous picture of an individual's performance[3], allowing analysis of behavior rate by reference to the slope of the curve[4]. We may see that the individual in Figure 3 initially scored twenty points by making a series of useful decisions which produced little expense to the patient. Later, however, additional decisions proved to be less appropriate, as well as very expensive, so that the simulation was completed with a cumulative score of about eighteen points and at a cost of $1500 to the patient.

Pathways: Expert Corridors. Individual pathway data are of most interest when viewed in relation to some standard. The plotting of data in the form of cumulative records allows construction of a corridor encompassing a range of acceptable or desirable performance for the particular simulation. Currently, such corridors are constructed in the following way:

1. The preferred pathway of the simulation author is plotted.
2. The pathways employed by the three review panelists (see above) are plotted.
3. The mean score of author and panelists is plotted at a number of points.
4. A corridor is plotted by charting the line defined by the points lying one standard deviation above and below the mean pathway.

An example of an author-panel expert corridor appears in Figure 5. The dimensions of the corridor are admittedly arbitrary, and are not intended to clearly delineate the only acceptable pathway(s). The corridor does serve, however, to facilitate comparison of individual pathways to a range of solutions effected by experienced clinicians.

Sample Data. A comparative plot for five participating dentists, with the corridor of Figure 5, appears in Figure 6. This figure shows a range of performance approximating to various degrees the slope of the suggested corridor. In Figure 6 we see a similar plot for a different simulation, showing a somewhat broader range of participant performance. It may be seen that increasingly cost-effective solutions are typically indicated by increasing slopes. Preliminary data indicate that this effect (increasing performance slopes) is seen across successive problems for an individual. That is, with experience in the simulations, participants tend to make better decisions and to become more cost-efficient in their solutions. An example of one participant's performance on four separate problems appears in Figure 7. The four plots are numbered chronologically (1-4), and are the first, second, fourth, and fifth problems completed by this individual.

These data are intended to be representative of the limited experience of the project to date, and to illustrate the approach taken in analysis. The project is now entering an active phase of participant involvement during which it is anticipated that additional data will be collected on approximately 300 dentist and student uses (i.e., 1800 individual problems).

POTENTIAL FOR CONTINUING EDUCATION. One of the questions we are addressing in our overall program evaluation is: Will the results of the project help meet needs in continuing education for dental practitioners?

Continuing Education Courses and Criteria. Continuing education courses in dentistry generally fall into the categories of practice management or clinically oriented content.[5] A few programs require active participation but most are of the lecture format. A majority of dentists read journals and articles in their own discipline and attend dental meetings as other means of continuing education. This is generally true of our participant pool.

A recent survey of dental continuing education needs in Nebraska revealed a need for a more personalized experience, and more active participation by the dentist[6]. Other recent articles[7,8,9,10] describe attitudes of dentists and needs in continuing education including a larger number of clinically oriented topics and better means of evaluating the effectiveness of various programs. The literature also indicates that such characteristics would make some continuing education programs, papers, and presentations more suitable than others.

Assessment of Simulations. In assessing the characteristics of the simulations, it would appear that the materials and the project activities meet the following criteria:

1. The simulations are clinically oriented.

The user must go through the simulation as s/he would see the patient in his/her office and make decisions about how to diagnose and treat the patient. The problems are based on real patients and many of them are unique problems not normally seen in general practice. The dentist can take some risks in the simulation which s/he would not take with a real patient.

2. The activity is interactive.

The participant must make decisions as s/he proceeds in order to complete the problem. Poor decisions may lead down an irreversible pathway, as would happen with a real patient. Participating dentists have made the comment, "It makes you think about what you are really doing." The simulations are designed to be as realistic as possible and can serve in lieu of actual participation in a clinical setting.

3. The learning experience is personalized.

The assignment of simulations is based on the individual's expressed needs and interest. Each assignment is unique.

4. There is built-in feedback within the materials.

The summary at the end of the lesson provides a complete statement of the correct treatment of the problem. Within the problem help sequences are available which provide information on which to base a decision.

5. The materials are based on broad national standards rather than on provincial standards.

Three panelists are brought in for 2½ days to review a group of 6-7 newly developed simulations. These panelists are selected for their broad experience in both general practice and dental education. The panelists review each problem singly and then as a group with the author, reaching a consensus about the ideal solution and acceptable alternatives. Additionally, each panelist is selected from a different geographic location to provide a range of views in background, education, and dental experience, resulting in quality materials. Three such panels have reviewed the completed simulations; two additional panels are scheduled in July 1979 and January 1980.

6. The problems are holistic in patient management.

The participant must view the patient as a complete individual in contrast to a crown or a denture patient. A holistic approach is not always used in the dental school curriculum and may need to be reinforced in dental practitioners.

7. The simulations provide a complete learning experience.

The library references and help sequences provide an opportunity to round out the simulation experience. The computer room is immediately adjacent to the library and learning center. Every effort is made to encourage the use of the library materials. Participants each receive a printed copy of the summary and references for each simulation they complete so that they can review and use this information at a later time. In addition, a special PLATO lesson is being designed on information access for health scientists and practitioners which includes strategies for information access, an introduction to sources, and a practicum in which the participant will have an opportunity to use the sources and strategies identified in the lesson.

8. The materials provide an opportunity to update knowledge and keep up with the expansion of new techniques and technology.

Many of the problems are unusual diagnostic problems for which treatment involves new techniques, medications, and materials. The references include articles on the latest research related to these problems. Since many of the participants have been in practice for a number of years, simulations can be an additional method of updating their knowledge. The special lesson described above also provides an update in accessing library information.

9. The material is of the caliber equivalent to a university level course.

Comprehensive problem-solving and decision-making activities are not generally included in the dental curriculum. The demands of analysis, synthesis, and judgment required by the complex simulation activity place these materials in the higher levels of cognitive skills.

10. The materials provide an opportunity for self-evaluation and overall evaluation of learning.

Each participant can evaluate his or her own approach to the problem by examining the data, reviewing the decisions made and the pathways selected. In addition, s/he can compare his or her pathways against those of the author, and if s/he chooses, with those of the panel members. Debriefing sessions are available with the author and with groups of other dentists.

After experiencing several simulations and reviewing the data, participants can see changes and improvement over a number of simulations and can recycle any number of problems for the purpose of improving decision-making and problem-solving capabilities.

At this point, we have not yet made final a credit system for continuing education units but it is our expectation that we will be able to arrange credit in the near future. Individual comments and positive support from participants indicate that this method of continued learning will be well received by dental practitioners. Finally, the general consensus of all panelists so far is that the simulations can be an important, effective teaching tool at all levels of dental education.

SUMMARY. The problems inherent in teaching and evaluating professional judgment are long-standing in dental education. Simulations can offer realistic situations in which to make decisions at minimum risk. Computerized simulations, with relevant library materials, are used as special resources at the University of Nebraska College of Dentistry as an innovative way to provide continuing professional education for dental practitioners and for advanced students who need experience in problem-solving activities.

Self-assessment data are collected in a professional experience profile and translated into an individual profile for determining the appropriate assignment of learning experiences that match the individual's needs and interest. Participants complete five to seven simulations ranging from acute to chronic, typical to unusual, with a range of treatment including major clinical dental areas.

Participants are oriented before beginning their assigned simulations, and are debriefed via problem summaries after each simulation. Individual pathways are charted in several ways to determine changes in problem-solving abilities. Trial use and demonstrations indicate a broad acceptance by dentists of this format. Review of continuing education needs and standards, as well as input from users, suggests that these simulations have much potential for continuing dental education.

REFERENCES

1. Stein, Leonard. Your Personal Learning Plan: A Handbook for Physicians. Illinois Council on Continuing Medical Education, Chicago, Ill., 1973.
2. McGuire, C.H., Solomon, L.M., and Bashook, P.G. Construction and Use of Written Simulations. New York, The Psychological Corporation, 1976.
3. Ferster, C.B. and Perrott, M.C. Behavior Principles. New York, Appleton-Century-Crofts, 1968.
4. Skinner, B.F. The Behavior of Organisms. New York, Appleton-Century-Crofts, 1938.
5. Russell, F., et al. "The Tri-State Program: A New Dimension in Continuing Dental Education." General Dentistry, 26:24-26, July-August 1978.
6. Miller, Elmer H. The Identification of Continuing Education Needs for Practicing Dentists in Nebraska. Unpublished thesis, 1972, University of Nebraska.
7. Cafferata, Gail, et al. "Continuing Education: Attitudes, Interests, and Experiences of Practicing Dentists." Journal of Dental Education, 39:794-800, 1975.
8. Chambers, David W., et al. "An Investigation of Behavior Change in Continuing Dental Education." Journal of Dental Education, 40:546-551, 1976.
9. Bauer, J. and Bush, R. "Dentists Attitudes toward Continuing Dental Education: Nontopic Factors of Demand for Courses." Journal of Dental Education, 42:623-626, 1978.
10. Milgrom, Peter. "Continuing Education and the Prospect for a National Standard of Dental Care." Journal of Dental Education, 38:482-486, 1974.

	OCCLUSION	ENDODONTICS	FIXED PROSTHODONTICS	OPERATIVE DENTISTRY	ORAL SURGERY	ORTHODONTICS	PEDODONTICS	PERIODONTICS	REMOVABLE PROSTHODONTICS
Occurrence Ranking	3.5	1.5	4.5	5.0	4.0	2.5	1.0	5.0	2.0
Referral Ranking		5.0				4.5		4.0	
Improvement	2	2	2	2	3	4	3	4	2
Interest	4	3	4	2	3	2	2	3	4
Data Gathering Abilities	1	2	2	2	3	4	4	4	2
Treatment Abilities	1	2	2	1	3	4	4	4	2
TOTAL	11.5	15.5	14.5	12	16	21	14	24	12

Figure 1. Sample Individual Profile

	Occ	Endo	Fixed	O.D.	Oral Surg	Ortho	Pedo	Perio	Remov	Syst-emic
U - C Neb 1	S	S			M					S
T - C Neb 2		S			S			M		
T - C Neb 3		M						S		
T - C Neb 4	M	S								
U - A Neb 5		S			M		S			
U - A Neb 6		S			S		M Diag			
U - A Neb 7					S					M Diag
U - A Neb 8	S	S			S			S		M Diag

Figure 2. Classification of Simulations

PLATO PROJECT

SCORING CRITERIA

+4	Critical to successful treatment (absolutely essential to optimal solution)
+3	Very Important (essential to any satisfactory solution)
+2	Helpful to solution, but not essential
+1	Appropriate or accepted procedure, but of minimal assistance to solution
0	Neutral, Optional (unlikely to help at this time, but not costly or risky)
-1	Inappropriate at this stage of this problem
-2	Harmful or unjustifiably costly (not irreversible)
-3	Irreversible treatment mistake (not life-threatening)
-4	Critical to health of patient (life-threatening)

Figure 3. Scoring criteria for decisions in Nebraska dental simulations.

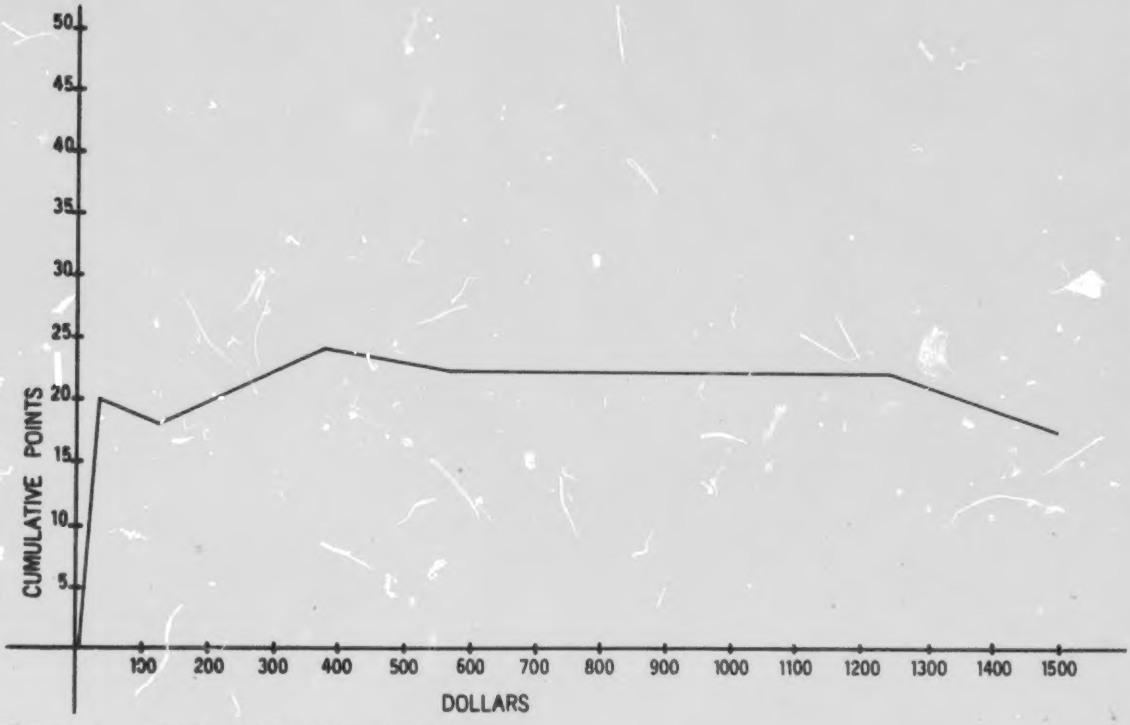


Figure 4. Sample cumulative record

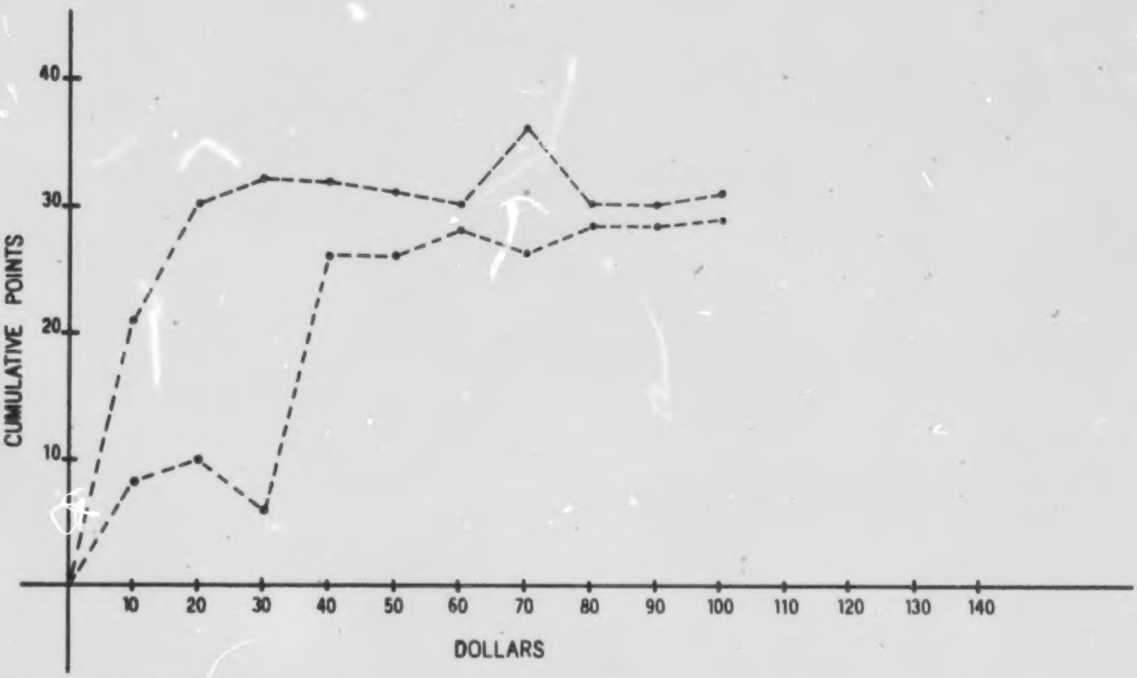


Figure 5. Sample "expert" corridor from Nebraska dental simulations.

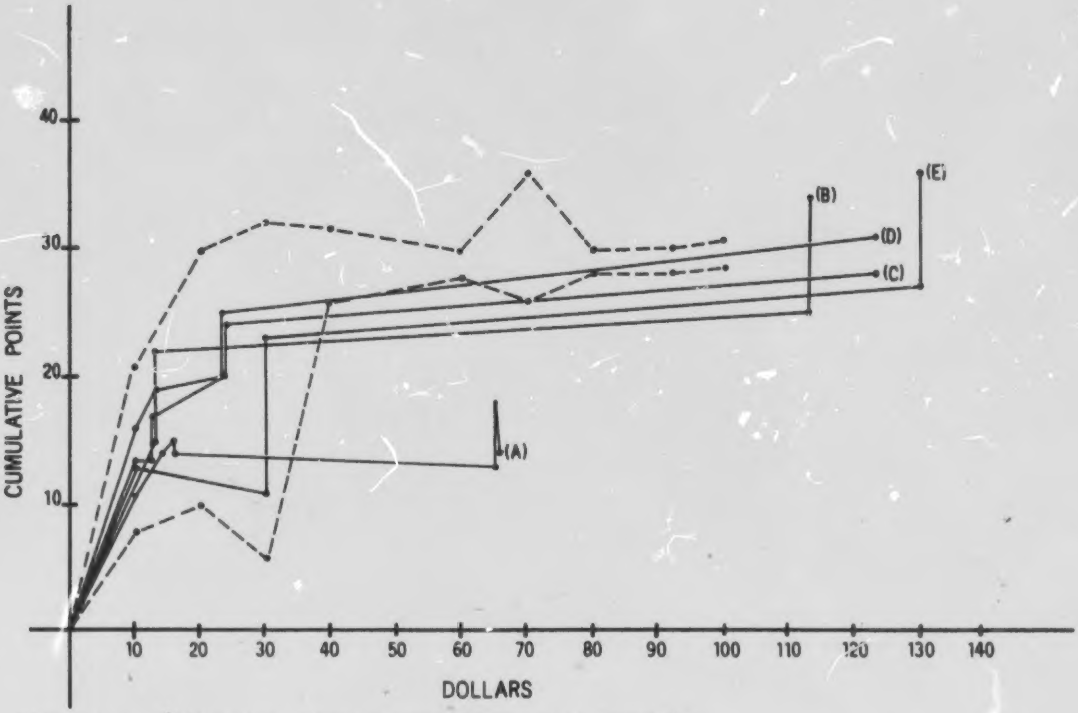


Figure 6. Individual participant data, with corridor.

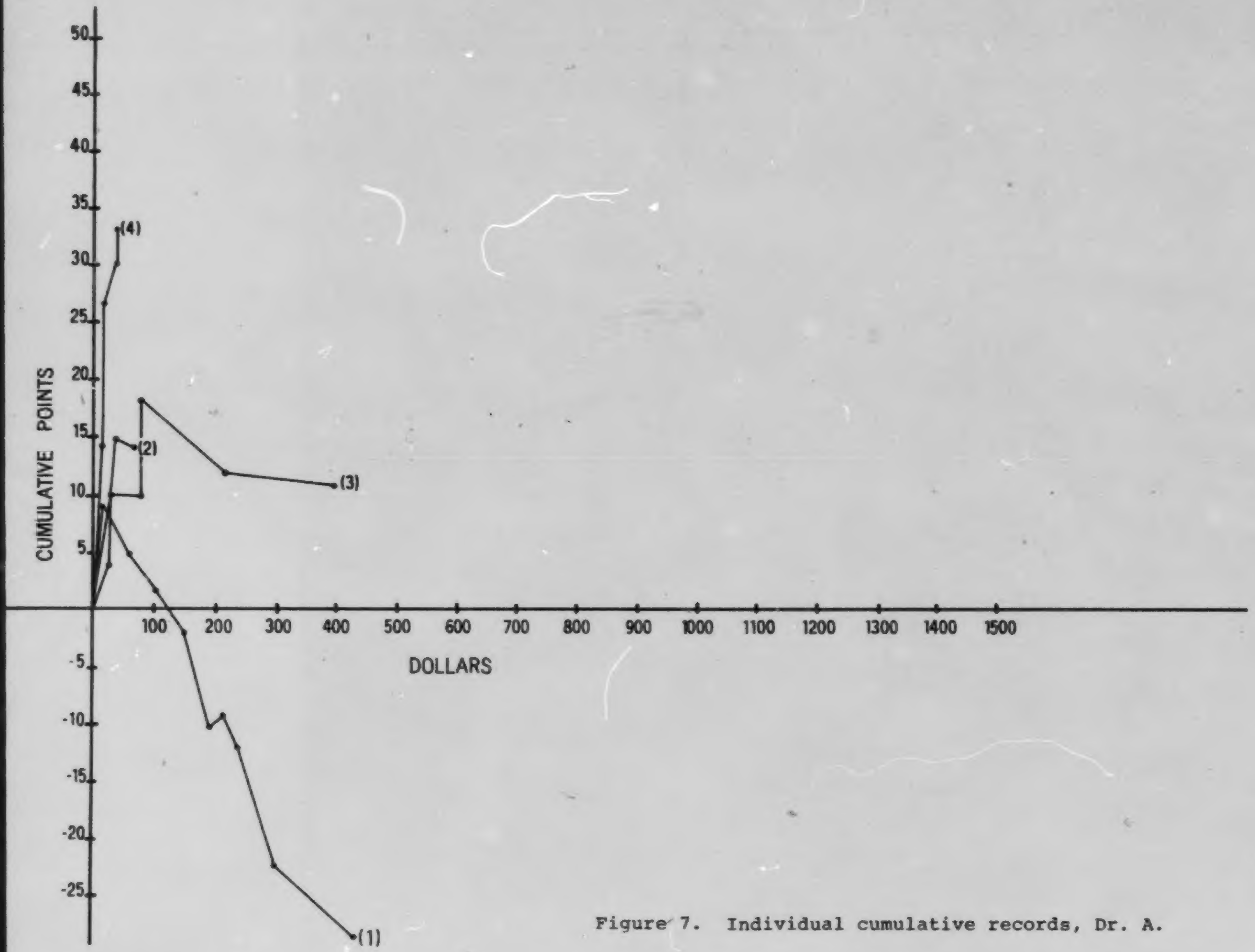


Figure 7. Individual cumulative records, Dr. A.

315

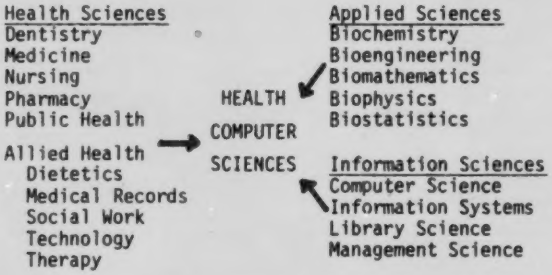
A TRAINING PROGRAM IN HEALTH COMPUTER SCIENCES[1]

Lynda B. M. Ellis, Ph.D. and Laël C. Gatewood, Ph.D.
Division of Health Computer Sciences
University of Minnesota
Box 511 Mayo Memorial Building
420 Delaware Street SE
Minneapolis, MN 55455

The training program in health computer sciences at the University of Minnesota is administered by the Division of Health Computer Sciences in the Department of Laboratory Medicine and Pathology of the Medical School. For over a decade, this program has provided graduate and postdoctoral training in the use of computers in biology and medicine to the students in the Health Sciences Center, including the schools of Dentistry, Medicine, Nursing, Pharmacy and Public Health, as well as the University Hospitals. Since 1974, the program has been supported in part by a National Library of Medicine (NLM) Training Grant. This paper describes the training program and discusses its accomplishments and projections for the future. It is hoped that such an exchange of ideas will help promote computing education in the health sciences.

I. Overview

The instructional model used by Health Computer Sciences can be illustrated by the following diagram. Students may come from one science area and need training in the other two in order to obtain and use these interdisciplinary skills. Thus the associate faculty must cover all three areas to provide the necessary background and direct research studies.



The educational goals of this training program are summarized below:

1) Instruct students in the basic discipline of computing for the health sciences.
2) Add advanced training in information, engineering, and computer sciences.
3) Provide graduate courses in health service informatics for all health professionals.
4) Find student research opportunities combining previous and acquired skills to promote health care.
5) Integrate coursework, seminars, consulting, and research into a comprehensive graduate program.

The following sections describe the faculty, students, courses, research opportunities, and supplementary materials which make up this program.

A. Faculty

The Program in Health Computer Sciences is directed toward computer applications for biomedical research, health services, and mathematical biology. Its faculty members teach classes, advise students and also conduct research in their specific areas of expertise.[2] These faculty hold graduate appointments in programs within schools of public health, medicine, pharmacy, and technology. This helps attract new students, provide input in curriculum planning, and give a broader base to the teaching program. Liaison faculty in other departments and affiliated hospitals also contribute to the training program. These include staff members from Biometry, Laboratory Medicine, Radiology, Surgery, Computer Science, Clinical Engineering, Management Sciences, Chemical Engineering,

---

[1] Supported in part by NIH grant LM-00160.

[2] Faculty in health computer sciences:
Prof. Eugene Ackerman - Health Sciences Computing, Mathematical Biology
Prof. Donald McQuarrie - Cardiovascular Surgery, Health Services Computing
Assoc. Prof. Stanley Finkelstein - Signal Processing, Biomedical Engineering
Assoc. Prof. Laël Gatewood - Health Information Systems, Quality Assurance
Asst. Prof. Myra Chern - Population Genetics, Medical Decision-Making
Asst. Prof. Donald Connelly - Clinical Pathology, Medical Support Systems
Asst. Prof. Lynda Ellis - Biochemical Simulation, Computer-Aided Instruction
Inst. Larry Sandler - Clinical Engineering, Biomedical Instrumentation
Res. Fel. Alan Moore - Electrical Engineering, Laboratory Instrumentation

Physical Medicine, and Pediatrics.

B. Students

As shown in Table I, a wide range of health science students from a variety of graduate programs are enrolled in courses in health computer science.

Table I

Graduate Programs at the University of Minnesota Which Use the Health Computer Sciences Training Program

Level of Use	Graduate Program
Post-doctorate Studies:	Residency in Clinical Pathology Mathematical Biology
Graduate Major:	Biometry and Health Information Systems Computer Science Biomedical Engineering Biophysics
Graduate Minor:	Biochemistry, Pharmacology, Physiology, Epidemiology Environmental Health
Supporting Field:	Social and Administrative Pharmacy Hospital and Health Care Administration Public Health Administration Laboratory Medicine Interdisciplinary Studies in Public Health
Required Rotation:	Residency in Laboratory Medicine and Pathology
Suggested Electives:	Public Health Nursing Public Health Nutrition Dental Public Health Veterinary Public Health Medical Technology Medicine, Nursing, Pharmacy, Dentistry

The level of student involvement ranges from 2-6 credit electives during professional training, a year sequence in biomedical computing during graduate school (12 credits), a full graduate minor or major (20 or more credits), to selected post-doctoral studies. Courses are described in the following section, which is divided into coursework for applied majors, supporting programs, or special emphasis classes for a health professional.

C. Courses

1. Core courses

Seven courses are considered core because they form part of the core requirements for computing-emphasis students enrolled in the program Biometry and Health Information Systems (BHIS). These courses fall into three main groups, a year (three-quarter) sequence in Biomedical Computing, followed by a second such sequence in Computer Methodology in the Delivery of Health Care, and a one-quarter course in Computer Consulting.

The Biomedical Computing sequence, offered every year, is required for all BHIS students and is also a service sequence. Its usefulness for students in other disciplines in Public Health (such as Epidemiology or Environmental Health) or Medicine (such as Physiology or Microbiology) as well as non-degree candidates, means that the majority of students enrolled in it are non-BHIS majors.

The first quarter of this sequence takes students with little or no prior-experience with computing or biology and medicine, and introduces them to interactive applications of computers in health fields. Examples are demonstrated both as algorithms and actual working programs in BASIC. The second quarter assumes a background in health terminology and algorithmic thinking. It stresses batch applications such as text manipulation, file handling, and the use of statistical package programs. The third quarter attempts to provide a perspective on a variety of more complex computer systems useful in biology and medicine. There are written assignments which require critical evaluation of existing programs, tours to local systems, and a term programming project which supplements the programming assignments found in all three quarters.

Once the student understands computing in the health sciences, a number of graduate courses are then available to extend the knowledge of computer applications. The Computer Methodology sequence has as a prerequisite the Biomedical Computing sequence. Thus it is assumed that students can program in at least one general purpose high-level language. have had an introduction to special purpose languages, and have a broad understanding of computer applications in health fields. The three quarters of this sequence are more loosely tied to each other than the preceding sequence. These courses are offered yearly, primarily for graduate students in BHIS, but also include students of Computer Science, Physiology, Biomedical Engineering, and related fields.

The first quarter introduces physiological monitoring and testing. The second quarter explores Bayesian logic, decision rules, and the computer's role in medical decision-making. The third quarter summarizes past, present, and future prospects for health information systems, with an in-depth evaluation of working systems as well as a discussion of their social, legal, and ethical implications.

The Computer Consulting course is offered every other year. It introduces the students to health service clients who describe problems that will become the class projects for the quarter. The lectures include the use of systems analysis and draw together the computer, engineering, and statistical techniques of the early course sequences in specifying objectives, system design, development schedules, and evaluation criteria for the project. The students work with the clients in preparing a proposal for institutional or governmental funding, and often continue consulting with the health team after the course is completed.

2. Recommended courses

Another set of courses offered in alternate years is not required for a Master's degree but is recommended for graduate students, especially those in Biometry and Health Information Systems. This set includes the one-quarter Analytical Techniques for Health Delivery Systems and a three-quarter year-long sequence in Mathematical Biology. The former course applies operations research and system analysis techniques to medical service systems. There is a special emphasis on applications of linear programming, queuing theory, inventory systems, and developmental planning models.

Each quarter of the Mathematical Biology sequence has two major emphases. The first quarter course includes discussions of the goals of mathematical biology and of deterministic models. The middle quarter emphasizes the use of mathematical transforms in describing transfer functions and the analysis and diagnostic interpretation of electrocardiograms and electroencephalograms. The final quarter includes information theory and its applications in genetics, as well as discussions of populations and stochastic epidemic models. Throughout, emphasis is placed on the design, selection and interpretation of models described in a computer-programmable fashion.

Finally, there is an Advanced Topics course which is offered to trainees and interested students, faculty, and staff. This course develops around special research interests and student requests. In the past, topics have included computer system design for the health sciences, small computer concepts and use, computers for clinical services, biomedical image processing, and pattern recognition and classification. All presentations discuss techniques and incorporate actual examples or case studies from the health sciences.

3. Courses for health professionals

The above courses were offered within the graduate program in Biometry and Health Information Systems. There is another entirely separate grouping of graduate classes which is offered by Health Computer Sciences to the health science community at large. These range from the weekly Joint Biophysical Sciences Seminars to general computer applications courses and special purpose courses in technology. The Seminar Series has a weekly attendance of 20-40 persons and presents investigators from the Health Sciences who describe their research and their use of quantitative methodologies.

An introductory computer applications course is given separately for health care providers and for health service administrators. Graduate students in professional and administrative programs have requested more exposure to technology and systems analysis in order to adapt to today's changing health care needs and environment. Computer concepts are illustrated through lectures, tours, readings, computer examples, and BASIC programming problems. These one-quarter survey classes are well-attended and are often recommended as a useful elective to students in related disciplines.

In addition to the general courses offered by the Health Computer Sciences faculty, a number of courses specifically tailored to students in Laboratory Medicine and Pathology are presented. All residents in the Laboratory Medicine and Pathology program participate in Introduction to Laboratory Computer Systems, a one-week intensive introduction to function, selection, operation, and management of laboratory computer systems. During this week residents are also introduced to the computer as a research and education tool.

A second course specifically designed for students of Laboratory Medicine and Pathology is an electronics instrumentation course aimed at medical technologists and laboratory researchers. This course concentrates on the electronics of analog instrumentation and uses department laboratory facilities and equipment. After an introduction to basic electronics via lectures and laboratory sessions, such common laboratory electronic systems as graphic recorders, pH meters, spectrophotometers, autoanalyzers, gas chromatographs, hematology analyzers, gamma and beta counters, and computer-based laboratory information systems are studied in detail.

D. Research Opportunities

Each student studying for a graduate degree is encouraged to participate in research activities. Master's students must spend 120 hours on one or more projects which are then documented in preparation for their final oral examination. These studies may lead to a doctoral dissertation, and some also provide funding, publications, or a continued job once the training program is finished. Any health science area is available for research, including clinical information systems, physiological testing, laboratory medicine, clinical trials, evaluation methods, and system modeling, among others. The University of Minnesota Hospitals and its affiliated hospitals in St. Paul and Minneapolis offer a number of sites for basic biomedical, health services, and epidemiological research. In addition, there are a number of health maintenance organizations, state health agencies, and health-related companies which provide field experiences for those students needing a practical emphasis as part of their training. An evening seminar is held every month to allow discussion of new research avenues by the students, faculty, and interested colleagues.

E. Supplementary Materials

As these courses have matured, accompanying materials and reports have also grown. All instructors must prepare instructional objectives, descriptive syllabi, and evaluation criteria for each quarter's classes. Four of the large service-oriented courses have accompanying workbooks or collated readings. A programmed instructional manual written to introduce our novice students to the BASIC programming language is also used as a resource by the Minnesota Educational Computer Consortium. A series of computer-assisted instruction (CAI) programs for health sciences use has been prepared and documented. Two major textbooks designed to accompany specific courses have been published. Each student must document a research project performed for graduate credit; these reports are filed and can be reproduced upon request. Every year the faculty and student presentations and publications are annotated and a bibliography prepared to share with others.

II. Acceptance

A. Student Participation

The National Library of Medicine sponsors ten training programs for Computers in the Health Sciences.[3] One of these was initially awarded to Minnesota in 1974 and has been renewed through 1983-4. The original impetus for the national training program was the recognition that traditional methods of medical education and health care delivery were changing to use new technologies and delivery systems. Therefore it was important to develop personnel skills for computer applications in medicine. This effort was aimed at teachers or potential faculty members in the health sciences, who could in turn train and influence others at their institutions. This grant served to formalize our training program which had been initiated in 1968 on an informal basis.

Our NLM training grant has supported seven post-doctoral and eight pre-doctoral trainees since 1974. Institutional and research funds have been found to support additional years at Minnesota for these trainees, thus allowing more students to be awarded traineeships. Five trainees obtained an M.S. degree and one the Ph.D. degree at Minnesota; two of these were subsequently Board-certified in Clinical Pathology. Eight trainees are currently enrolled in the program. All previous trainees are currently associated with academic medicine and health services or are making satisfactory progress toward a graduate degree at another institution.

The fifteen official trainees reflect only a portion of the students who have been advised by the faculty of Health Computer Sciences since 1974. Five other physicians have been trained in this program; these students have received two Ph.D. and three Master's degrees in Biometry and Health Information Systems. All but one of these previous students are involved with academic medicine; one is affiliated with a biomedical systems company. Currently there are three other physicians and one mathematician receiving post-doctoral training under the direction of our faculty.

There are also many other pre-doctoral students in Biometry and Health Information Systems, Public Health, Laboratory Medicine, Computer Science, Biophysics, Bioengineering, and Administrative Pharmacy for whom the faculty in Health Computer Sciences have served as graduate and research advisors. Thirteen M.S. students emphasized biocomputing, and eleven emphasized biostatistics in Biometry and Health Information Systems; four M.S. students were from other graduate programs, and one student received a Ph.D. in Biophysics. Currently there are fifteen other students working on graduate degrees that involve Health Computer Sciences.

B. Program Review

Criteria such as support requests, employer demand, and trainee success indicate the value of the program in training scientisits for academic environments. A new type of health scientist with interdisciplinary skills in statistics, computing and simulation is being created to work with other health professionals in addressing problems of health delivery systems. These persons can review difficulties impeding information flow, recognize those amenable to computer solution, translate system requirements for computer consultants, and coordinate the resultant services.

The operation of this graduate training program for the last five years has indicated areas of maturation and growth which required additional efforts to continue program cohesiveness. Program resources could easily be extended to cover an additional group of trainees. The current requests for student training and program graduates indicate that doubling the number of trainees to ten each year would be appropriate, and we will do this as of this July. Another concern is communication between students from various specialty and graduate levels. These students represent a valuable resource to each other; conference space and research seminars are needed for closer interactions. A third concern is the place of CAI programs in the health sciences. Authors, sponsors, and reviewers should be found to keep material viable; materials could then be coordinated by Library Learning Centers as part of the instructional resource. Finally, there have been many requests for information from other institutions. Dissemination and sharing of the training description and program emphases is becoming an important function.

---

[3] Other NLM Training Grant institutions are:
Case Western Reserve University
Duke University
Georgia Institute of Technology
Mt. Sinai School of Medicine
Ohio State University
University of Alabama at Birmingham
University of California at San Francisco
University of Illinois at Urbana
University of Missouri

III. Future Directions

There are several deficiencies in the training program as currently implemented at Minnesota, which have been brought out by experience. First is that training is separated in several graduate programs. Although each one claims student credit and endorses the training, none are willing to continue operations without the current training grant support. A permanent University base must be found for this interdisciplinary type of graduate study. Support must include commitments for faculty, trainee expenses, graduate assistants, and preparation of instructional materials.

The second problem is that although the courses are well integrated into Laboratory Medicine, Public Health and Pharmacy, they have only begun to affect curriculum planning in Nursing, Dentistry, and the rest of the Medical School. Introductory sequences must be available at appropriate times of the year to fit training in health informatics into these highly scheduled programs. These courses should adjust prerequisites and objectives to serve as bridges for health professionals wanting to emphasize computing for administrative or research purposes.

A third problem is that students need more exposure to bioengineering and information systems. Our students have been well trained in biostatistics and biocomputing, but need more training in several areas, including digital instrumentation, communication networks, data base management systems, and microprocessors. This entails specific collaboration with other graduate programs currently at Minnesota. Affiliation with the Bioengineering program has already begun. The relationship to the Management Information Systems program, including cross-sharing of students and courses, remains to be developed.

The fourth problem reflects the changing attitude toward the delivery of health services. The emphasis is switching from instituional (i.e. in-hospital) care to ambulatory and other forms of community care. In addition, the state health agencies and even health industries are becoming involved in outreach services, certification and licensure, and clinical monitoring. The students have been obtaining research experience as new projects were initiated within the health sciences. As valuable as this research exposure is, it does not emulate actual daily experiences with health service problems. There, the students should be obtaining more consulting experience in project design, proposal preparation, service evaluation, and personnel coordination. A required field internship may help establish these types of skills, and point the students towards appropriate areas for health services research.

Finally, we also plan to begin a graduate program in information systems for the health service professional. These persons would serve as change agents and coordinators in establishing new uses of computer and system technology within their health specialty. Clinical decision-making, as well as service management, must be integrated with current health information systems. The provision of better health services through more systematic uses of biotechnology would result from augmented professional education.

IV. Summary

The training program in Health Computer Sciences has been initiated successfully and has future goals of enrichment, specific strengthening in several areas and an additional emphasis in training the health service professional. The next five years should establish the utility and role of the training program as a whole at Minnesota. Each of our goals can be examined in terms of its effect upon the graduate students in Health Computer Sciences. If an objective is not achievable with reasonable effort, then it may not be an appropriate part of training and should not be pursued.

We have a full graduate program training different levels of health service professionals. Studies can lead to a M.S., M.P.H., or Ph.D. degree and can include a clinical residency. These uniquely trained health professionals will teach, coordinate, and develop appropriate biotechnologies to help contain costs, increase access, and improve the quality of health care.

DEVELOPING A TEACHING/LEARNING EXPERIENCE FOR NURSES IN FUNDAMENTALS OF COMPUTER PROGRAMMING PRELIMINARY TO NURSING RESEARCH

Kathryn Erat
Research Associate
Department of Surgery - Computer Science

Susan McGrath
Assistant Director of Nursing for Staff Education
Peter Bent Brigham Hospital
A Division of the Affiliated Hospitals

Awareness of the continued growth of the Health Care Industry, its connection with computer technology, and their impact on nursing motivated us to assess the nurses' need for computer knowledge. Our initial step was to review the literature and survey nurses in our agency to determine the program activity we would pursue.

Why Computer Knowledge?

Much has been written about the need for research in the nursing profession. Nursing leaders in education and service agree that the basic nursing curriculum for preparing the professional nurse must include course content that encourages the learner to develop skills in research process and methodology to prepare for his/her role on today's health care team. We believe that all nurses participate in nursing research although some are in more structured roles than others. Specifically, the nurse as a researcher should be conversant with good data collection methods, analysis of data (statistics), and the support computerized data base systems and statistical computing packages can give to nursing practice and nursing research.

The nurse educator needs to be aware of the potential for computer-aided instruction in nursing education -- not only on the baccalaureate and graduate level in a university setting but also for continuing education outside the university. For example, in the rural areas computers could provide programs of the same level of sophistication as those in the large academic community. A computer system could provide instruction, continuing evaluation, research, and record keeping.

There are many technical, financial, and administrative problems to be resolved and political and ethical questions to be answered before this method of continuing education becomes wide spread. This method should be demonstrated to the educators and practitioners who would recognize the benefits to health agencies, the consumer, the clinician, and the profession. The barriers are being explored and means to deal with them are being identified. Some progress is being made in the implementation of these concepts.

The professional nurse should be prepared to deal with computer technology as it relates to decisions of professional nursing practice and the health care industry if nurses are to assume significant roles as health care providers.

It is reasonable to require that the largest group of health providers (nurses) be aware of the structure and capabilities of computer-based health information systems. For example, much health agency collected and disseminated data are computer based. An appreciation of the technical structure of the system will enable the nurse to have input into the system and to respond appropriately to the feedback from the system.

DISCERNING BASIC OBJECTIVES FOR A COMPUTER COURSE FOR NURSES

We know that many of the nurses in middle management and staff positions in health agencies large and small have not had training in basic computer use. How do we help them? How do we prepare them to function in today's computer environment? How can we help them to respond appropriately to this computer-oriented health care system?

The remainder of this paper will describe an attempt to answer the above questions and the development of a learning/teaching experience in "Fundamentals of Computer Programming for Nurses" for nurses in our agency -- the Peter Bent Brigham Hospital, a Division of the Affiliated Hospitals Center, a 330-bed adult, acute medical-surgical teaching hospital in a large metropolitan area. This program (see Appendix I) was approved by the Massachusetts Nurses Association for 1.2 Continuing Education Units.

Of primary concern to us was the identification of need. How does the nursing staff view computer science and programming? Do they see a need for computer science in their practice? How can we help them recognize that computer knowledge is significant for quality nursing and help them to know who needs to learn about computers.

We learned that the nurses were aware of these needs and identified specific areas of application.

1. Administrators

   Those nurses -- Directors and their Assistants, Supervisors, Head Nurses, etc. -- involved in decision making relative to development of information systems need a knowledge of computer technology. They recognize that in the ever-enlarging scope of nursing, they need the assistance provided by computer systems to establish standards of nursing practice.

   Two examples of effective use of computers are:
   a) Staffing time schedules
   b) Scheduling in Operation Room

   These have been proven efficient and cost effective.

2. Educators

   This group needs to be aware of the power and method of using the many automated medical bibliographic retrieval services.

   They also need to recognize the availability and methods for analyzing public domain health data bases which highlight the health and morbidity trends that nursing education should address.

   They need to be able to impart to their students an awareness of computer potential.

3. Staff Nurses

   All nurses who use monitering equipment that incorporates micro/mini computers must know computer basics in order to correctly analyze data and make appropriate judgements regarding instrument reliability and accuracy and patient care.

There are other groups, but these three include a very large segment of the nurse population who have a need for basic computer knowledge. The need is clear.

Now, where shall we start? At what level will we begin to motivate these nurses to move into this area of learning?

We found that a few nurses had good research backgrounds and had participated in or completed one or more research projects.

We learned that some of the nurses who had Research & Statistics in their academic programs treated it as a necessary evil with little application to nursing. They suggested that little internalization of this knowledge had occurred, and texts, notes, and knowledge were quickly set aside and with disuse quickly forgotten.

We also learned that other nurses have no research or computer background at all. Therefore, our starting point was established -- we would begin at square one.

In developing the objectives for this course, we were influenced by (1) the awareness that generally there has been limited application of research process in nursing practice, and (2) the demands on the nursing staff by their assigned role. Nursing is a full-time job which leaves little time and energy for assigned readings and the designing and implementing of computer programs. Therefore, the objectives must be realistic to nursing practice and within the realm of achievement in the time frame established.

These considerations led us to adopt an andragogical[1] approach. This approach dictates that content would be developed around a clinical focus with immediate application which demonstrates results. Malcolm Knowles tells us that, "Adults will learn that which is applicable to the work setting."[1] It is well to let the learner establish the focus of application and select the subject of his/her study, the program will provide him/her with the opportunity to learn basic knowledge and skills needed to complete the study. This application of an adult education principle endorses the old Chinese proverb[2], "I hear and I forget. I see and I remember. I do and I understand."

With this phase completed, the basic design began to take shape.

COURSE GOALS

The next task was to decide what aspects of computer knowledge nurses should know and how to teach that knowledge. The goals were:

1. To provide the nurses with a broad introduction to the principles of how computers work, how these principles have been applied both in hardware and software, and how hardware and software have been organized into different system architectures and configurations to do meaningful work in various real-life applications.
2. To introduce the nurses to the kinds of analyses one applies to a given problem or situation when one is considering using a computer.
3. To help the nurses acquire a limited but reasonable fluency with the vocabulary of computer science.
4. To introduce nurses to the specific use of the computer in analyzing research data. This introduction would include forms design, data coding and collection, the use of canned statistical programs, and the construction of simple programs of analysis.
5. To insure that at the conclusion of the program a participant would have a well-organized complete body of knowledge that would be useful and adequate in his/her professional activities, and at the same time be a sound foundation for more advanced work if he/she found this necessary or desirable.

COURSE DESCRIPTION

We decided on the following program activities:

1. Explaining the components of a computer (CPU, peripherals, etc.)
2. Introducing the concepts of computer language and the basic techniques of programming both at machine and higher language levels.
3. Introducing the concepts of systems analysis in relation to estimating the quantity of data to be processed, the timeliness of the processing, and

appropriate coding schemes for data (Appendix 3).

4. Introducing the concepts of various system configurations (hardware and software) for different applications (data base, real-time, distributed systems, time-sharing, etc.)
5. Providing information concerning journals, texts, and societies so that students would know where to turn for further information.
6. Providing computer lab experience.

The constraints to devising the program to carry out the above activities were:

1. The program would have to be given during the normal working day. This meant that the attendees would have to be excused from normal assignments or perhaps plan to attend their program on their days off. (With nurses working rotating shifts, it is very difficult to schedule a several hour course at times that are convenient to everyone or at times when everyone is normally present). It was decided that the maximum amount of program time would be 12 hours of lecture plus optional lab sessions of as many hours as the participant found practical. The program met two hours per week for six weeks. Six weeks seem to be an adequate time for this new material to coalesce and to allow sufficient time for each participant to schedule several hours of supervised and unsupervised lab.
2. The attendees would have varying backgrounds of science and math and cover the age range of twenty to sixty; some would be recent students, others not.
3. The program would be presenting materials not previously encountered by most participants and would be demanding thought processes and learning activities quite different from the usual nursing curricula and learning techniques. Since medicine and nursing are both arts as well as sciences and since patients are more often highly individualistic than classic text-book cases, the practitioners of both medicine and nursing have to become comfortable at an early stage in their careers with ambiguity and the concepts of testing, trying, evaluating, and adjusting techniques. Because most of what they do to patients is highly physical, a premium is put on interpersonal skills and attentive observation. Other skills, though their rationale is scientifically established and rigorously taught, are highly developed by clinical observation and imitation. This learning and performance gestalt appears to be somewhat antithetical to the typical computer science gestalt where a premium is put on abstraction and careful, complete, totally contained analysis before implementation. Computer science demands that ambiguity be resolved satisfactorily rather than tolerated (all paths in the flow-chart must end somewhere in a meaningful activity). The activity of programming is a rather lonely pursuit with minimal physical activity, and computer feedback is a poor substitute for interpersonal interaction.
4. Most of the attendees would be women. This raised the possibility that well-documented[3] math anxiety would manifest itself among the attendees because of the similarity of techniques and abstraction between computer science and math.
5. This was to be an adult education program where absorbing principles and skill mastery to a level satisfactory to the attendee were to be the criteria of success, rather than a grade from the instructors. The attendees had many competing demands on their time, and the requirements for study outside of class and lab had to be very flexible. Supervised lab sessions of one hour each for one to three participants were scheduled frequently; attendees were allowed to sign up for several. The computer was available unsupervised at almost any other time for anyone who wanted to use it.

Course materials consisted of:

1. A 40-page set of notes setting forth basic computer principles and introducing assembly language programming in a coherent progressive structure.[4]
2. A vocabulary list (Appendix 4) organized to reinforce the structure and relationships of the concepts being taught.
3. Texts on FORTRAN, some terse, some elaborate.[5,6,7,8] FORTRAN was the high-level language chosen because of its ubiquitousness. A Hewlett-Packard 2116B computer was available for hands-on programming.
4. A resource collection of 24 articles from various journals elaborating on all phases of material presented in the course (Appendix 5).
5. A self-administered examination constructed to elucidate most of the principles and vocabulary covered in the program (Appendix 6).

The program was presented four times over an 18-month period. Each time between ten and sixteen persons participated. With one exception, there was at least one male (not a nurse) and one or two non-nurses in each program.

PROGRAM EVALUATION

The evaluation is based on the instructor's observation, the self-administered test (Appendix 6), and the student's written evaluation (Appendix 7). All attendees became comfortable with the idea of computers (they lost their mythical mystique), developed an appreciation of their potential, acquired ease with the vocabulary, and successfully completed the self-exam. Several attendees with research projects acquired the skill to code and organize their data for computer analysis. All attendees realized the advantages of seeking a computer specialist's advice in setting up a project involving data collection and manipulation.

It is assumed that several participants gained sufficient knowledge to be comfortable in interacting with systems analysts integrating computerization into an area of health-care for which they were responsible.

They all gained, if not some rudimentary programming ability, at least an appreciation or awareness of the techniques.

Practically speaking, it appears that the goal of imparting FORTRAN programming skills in so few lectures that were also covering other topics with limited outside work and a batch-oriented lab was for most participants too ambitious. A batch-oriented lab with a slow printer is just too inefficient. For the last group in the program, it was possible to switch to a time-shared BASIC lab which was much more efficient and much more productive. In that group all the participants successfully completed, in a very short time, an assignment to do ascending and descending sorts of random numbers.

This was felt to be a good introductory problem because it involved input/output, conditional branching, nested looping, and subscripted data arrays -- most of the standard programming techniques. The problem was not assigned until students were shown these techniques in machine language stressing address modification.

The instantaneous feedback of errors and successes in BASIC and the freedom to experiment interactively seem to by psychological reinforcers sufficent to encourage rapid learning in the absence of other pressures such as grades, assignments, etc.

The male attendees and those with other highly quantitative backgrounds did exceptionally well. They made much use of the computer outside of lab taking a let's conquer this attitude not exhibited by the nurses. Whether their aggressiveness in mastering the material is to be attributed to maleness, the teachers femaleness, more free time to devote to the program, previous confidence with quantitative analysis, or other goal imperatives that involved knowing computers is unknown.

Many of the nurses were restrained in lab by being concerned that they might push a wrong button, damage the computer, etc. Much instructor reassurance was required. Whether women are more conservative, less confident, more wary of the unknown than men or less likely to have used a computer is unknown. Many of the nurses when questioned about their restraint mentioned concern for equipment, reluctance to make demands on an instructor, or interrupt other students as reasons for their less assertive behavior.

The area of greatest difficulty was in helping those who were not comfortably intuitive in applying programming constructs to a problem. For example, given the problem of finding the largest number in a group, the answer is so obvious if one simply looks at the numbers, a perception depending heavily on sight and prior knowledge, that transference to a computer approach of devising a series of small, precise, repetitive steps where sight and specific prior knowledge about the numbers plays no part is difficult for some students while others just have no interest in this approach. The substitution of logical constructs for what is easily perceived by sight is not an easy task for some. We also conjecture that math anxiety may play some part in this difficulty. We would like to do more work on developing teaching techniques that help assist students who are not quantitatively oriented to make this transference.

Although one of the stated goals of the course was for the nurses to analyze a data set statistically, this proved too ambitious an assignment for some given the short duration of the course and the inefficiency of conducting a lab using FORTRAN. This project should probably be a follow-up workshop[5]. A crucial first step was achieved in this regard in that all students grasped the principles of form design, question formation, refinement of questions to eliminate ambiguity, coding schemes, and the necessity for pretesting data-collection protocols. They also understood that computers could do analysis even though nurses were unable to program the analysis at this stage. They will probably be comfortable in consulting with computer specialists who could do the analysis or give them further guidance.

RECOMMENDATIONS

The program is useful and successful. It is probably too ambitious. Whether the ambitiousness offers widened horizons, enhances confidence and encourages bold designs or intimidates, overwhelms, and discourages further attempts to learn more is probably dependent on each attendee. It would certainly be a plus if some of the nurses blossomed into computer experts; but since most seem to be attracted to nursing because of the highly interpersonal activity of the profession, it is reasonable to assume that their interest in computers is an interest in just one of many scientific disciplines that they use as the need arises. Therefore, any program that adds to this scientific armamentarium is successful for the individual. For the nursing profession, it is probably imperative that more personnel in administrative positions become extremely comfortable with the material presented in this course. For that group, it is recommended that a one-week seminar be devoted to this type of material. During this time, the attendees would be freed from other responsibilities so that time could be spent in study as well as lecture and lab. This type of opportunity for computer exposure will probably become a reality as more large hospitals undertake ambitious computer projects impinging directly on patient care. More work needs to be done on the non-assertiveness of many women in acquiring quantitative and scientific principles and techniques and how to overcome this non-assertiveness of many women in acquiring quantitative and scientific principles and techniques and how to overcome this non-assertiveness[2]. More work needs to be done on how to teach problem analysis as an adjunct to programming. More work needs to be done on how to help older persons whose modes of thought have been exclusively organized by the domain of discourse of one profession for several years immerse themselves in new patterns of thought.

CONCLUSION

As we move toward a more highly developed health care system involving many disciplines, the nurse's awareness of the need for knowledge of computer technology will be the motivating force to move him/her toward more sophisticated experiences in this field.

The basic Computer Programming for Nurses was a hesitant first step to expose him/her to the world of computers. "Adults tend to learn in response to pressures they feel from current life problems," say the proponents of adult education[2]. Nurses are adults and many of the current problems that are affecting their work are the problems that are associated with or may be resolved by computer technology. When the pressure is great enough and the need is clear, this first step will help them move toward a level of sophistication necessary for them to function effectively in today's health care system.

BIBLIOGRAPHY

1. Knowles, Malcolm; The Modern Practice of Adult Education, New York, Association Press, 1970.
2. McGugin, Martha et al; "An Andragogical Approach to Teaching the Problem Oriented Method of Recording," Journal of Continuing Education in Nursing, Vol. 10 #1, pp 7-10.
3. Schafer, Alic, T.; "Brief Description of The Wellsley College Mathmatics Project." A Component of Wellsley College -- Wesleyan University Mathematics Project, Wellsley, MA, September, 1978.
4. Erat, Kathryn; "Computer Seminar," In House Document Peter Bent Brigham Hospital, Boston, MA, 1964.
5. Hewlett Packard; FORTRAN IV Reference Manual.
6. Leeds, H.; Weinberg, G.; Computer Programming Fundamentals, McGraw Hill, New York, 1966.
7. IBM, Introduction to IBM Processing, IBM Corp., Poughkepsie, NY, 1965
8. McCracken, D.; A Guide to FORTRAN IV Programming, John Wiley & Sons, NY, 1972.
9. Sweeney, MaryAnne; Olivieri, Peter; "Driving Nurses to Research," Nurse Educator, July, August, 1978, Vol. III #4, pp 7-13.
10. Bendix, et al; "Computer Scheduling for the OR," Modern Health Care, June 1976, pp 16-169.
11. Collart, Marie; "Computer Assisted Instruction and the Teaching-Learning Process," Nursing Outlook, August 73, Vol. 21, #8, pp 527-532.
12. Cartwright, Steve; "Hospital Wide CC TV," Health Care Education, October/November 1978, pp 18-21.
13. Krueger, Janelle; "Utilization of Nursing Research: The Planning Process," Journal of Nursing Administration, January 1978, pp 6-9.
14. Porter, Sharon; "Application of Computer Assisted Instruction to Continuing Education in Nursing," Journal of Continuing Education in Nursing, November/December 1978, Vol. 9, #6 pp 5-7.
15. Schare, Barbara; "An Undergraduate Research Experience," Nursing Outlook, March 1977, pp 178-181.
16. Somers, June; "Information Systems: The process of Development," The Journal of Nursing Administration, January 1979, Vol. IX, #1, pp 53-58.
17. Sparks, Susan, M.; "Letting the Computer Do The Work," American Journal of Nursing, April 1978, pp 645-649.

APPENDIX I

I. Purpose

This program is designed to orient the nurse to the gestalt of the computer milieu in order that he/she may interact more knowledgeably with computer professionals in 1.) planning patient care systems that utilize computers and 2.) planning, implementing, and evaluating projects in nursing research that involve computers -- thereby improving patient care.

II. Objectives

At the completion of this 12-hour program, the nurse will be prepared to:

1. Code, score and scale data from Nursing studies, "Orientation of the New Graduate at Peter Bent Brigham Hospital" and "Patient Attitudes Regarding the Functions of the Nurse in an Extended Role."
2. Utilize FORTRAN in computer programming.
3. Identify the basic components of a computer system and their functions.
4. Utilize a basic vocabulary of computer terminology.

III. Format

A. General

1. Prior to this program, the nurse will have reviewed the steps of the Research Process.
2. This program, which incorporates the principles of Andragogy, will be flexible and proceed at a rate mutually determined by the instructor and learner.

NOTE: If additional time is needed to attain proficiency in computer programming, this may be arranged. This additional time will not be included in program time, will not affect CEU's, and will be off-duty time.

B. Specific

1. Didactic sessions and laboratory sessions of demonstrations and guided learning experiences will be the format of instruction.
2. The instructor-learner ratio in laboratory sessions will be 1 to 2.
3. Learners will come to class prepared, by reading assignments and/or computer assignments, to participate in class activity.
4. The focus of the learning experience will be two Nursing studies, "The Effect of the New Graduate Orientation at Peter Bent Brigham Hospital" and "Patient Attitudes Regarding the Functions of the Nurse in an Extneded Role." The nurses will critique these through the Research Process and use the data as the basis for learning computer programming, data processing, and analysis.
5. Laboratory time will be on hospital computers; on line access to HP2100A Real Time Operating System and HP2116B Disc Operating System.

IV. Participants

This program is designed for the professional nurse. Prerequisite: Exposure to research methodology.

V. Schedule of Classes

Classes will be held each Tuesday and Thursday, 1:00 to 2:00 pm, for six weeks. The first class will be held June 15, 1976.

IV. Evaluation

Evaluation is determined by the degree to which the learner achieves the course objectives.

This will be evidenced by:

1. Achieving an acceptable score on quiz (80 or above). Quiz will be used as a self-analysis of progress and identify learning needs.
2. Successful participation in constructing a computer program for data analysis related to the Nursing studies. This means he/she will code data and program the computer.

VII. Coordinator

Susan P. McGrath, Assistant Director, Staff Education.

VIII. Faculty

Kathryn Erat, Research Associate, Peter Bent Brigham Hospital; Patricia E. McGrath, Nursing Coordinator, Peter Bent Brigham Hospital; Mary Fay, Nursing Coordinator, Peter Pent Brigham Hospital; Susan P. McGrath, Assistant Director-Staff Education, Peter Bent Brigham Hospital.

IX. Assigned Tests

1. Hewlet Packard, FORTRAN IV Reference Manual.
2. Leeds, H.; Weinberg, G., Computer Programming Fundamentals, McGraw-Hill, (New York), 1966.
3. IBM, Introduction to IBM Processing, IBM Corp., (Poughkeepsie, New York), 1965.
4. McCracken, D., A Guide to FORTRAN IV Programming, John Wiley & Sons, (New York), 1972.

APPENDIX 2

COURSE OUTLINE

Class Content

I. Research in Nursing -- a Review

A. Process of Development
  1. Purpose
  2. Objectives
    a. Behavioral
  3. Philosophy of Education
    a. Training vs. Educating
    b. Andragogy vs. Pedagogy
  4. Evaluation
    a. Achievenent of Objectives

B. Review of Research Process
  1. Steps of Research Process
  2. Survey of Research Studies to be critiqued re: adherence to Research Standards.
    a. Exploratory Study of the Effect of New Graduate Orientation at PBBH.
    b. Exploratory Study of Patient Attitudes Regarding the Functions of the Nurse in an Extended Role.

II. Introduction to Computers

A. Binary Coding, Octal, Fixed and Floating Pt. Numbers
B. CPU Organization and Function
C. General Programming Concepts
D. Programming Languages
E. Memory
F. I/O

Lab Assignment

1. Introduction to computer
2. Disc loading

III. Forms Design

A. Coding Techniques
B. Question Structure
C. Mechanical Sorting

Lab Assignment

1. Data preparation for computer

IV. Problem Analysis

A. Choice of Assignment to One of Three Problems
  1. Form Design
  2. Data Preparation for Computer (paper tape punching & editing)
  3. Selection of Statistical Techniques Relative to Problem

V. Introduction to FORTRAN

A. Variables
B. Constants
C. Arrays
D. Dimensions
E. Arithmetic Statements
F. Control Statements
  1. Go to xx
  2. Go to (xx,....), N
  3. IF
G. Loops
  1. Do
  2. Continue
H. Input/Output
I. Subroutines

Lab Assignments

1. Writing of program for problem analysis
2. Computer editing, running & checking of program

VI. Discussion of Large Computer Systems
A. Real-Time
B. On-Line
C. Interactive
D. Time-Sharing
E. File-Structures
F. Terminals
G. Man-Machine Interface

VII. Analysis of Results of Program
A. Evaluation of Computer Utilization Regarding Nursing Studies
B. Critique of Nursing Studies to Evaluate Adherence to Research Process
C. Evaluation of Program Regarding:
  1. Plans for Follow-Up
  2. Design for Further Study

APPENDIX 3

Necessary Viewpoints in Analyzing a Problem or Job

Problem -- Solution is sought; appropriate method must be selected or devised, e.g., a statistical test

Job -- A task or series of tasks that can be carried out by successive procedures.

A. Know the problem or job in itself.
  1. Analyze the problem or job. What is the desired end result? There might be several desired ends or results.
  2. What is the input, raw data?
  3. What process will turn the raw data or activators into the desired end results (an activator may be a signal, alarm, etc. rather than data to be transformed or an input to a function)?
  4. What is the output. How does the job relate to real world?

B. Think of input, output, and process as a system related to computer potential.
  1. Input device(s)
  2. Output device(s)
  3. Storage (capacity, speed, access, longevity).
  4. Retrieval
  5. Processing capability
  6. Timeliness
  7. Orderliness
  8. Redundancy
  9. Recovery
  10. Modification
  11. Intervention & control

C. Think of list under B in relation to available hardware.

D. Think of lists A & B in relation to available languages, operating systems, and canned procedures.

E. Describe data and its interrelationships. Organize rules (dictionary) for its identification and storage.

F. Translate tasks defined by the steps in A into procedures to be executed in a logical order.

G. 1. Establish hierarchy of tasks.
  2. Extract features of generality for tasks.
  3. Establish rules for communicating among tasks. Tasks will fall into one of the following:
    a. Directly related to logical processing of job.
    b. Required to embed job in a particular language.
    c. System design or operating environment.
    d. Required to meet standards established by list B.
    e. A task that many other tasks find necessary or useful.

H. Establish all known options that can result from input or processing.

I. Test system for accuracy.
  a. All branches
  b. Limiting cases on data
  c. Inaccurate data.

J. a. Document input, process, and output.
  b. Users' instructions

The human factors angle has been omitted in the above, however, it would be another criteria in list B if applicable.

BIBLIOGRAPHY

Abdellah, F.; Levine, E., Better Patient Care Through Nursing Research, MacMillan Company, (Toronto), 1970.

Hasse, Patricia; "Pathways to Practice, Part I," American Journal of Nursing, May 1976, pp 806-810.

Hewlett Packard, FORTRAN IV Reference Manual, Hewlett Packard Co., 1970.

IBM, Introduction to IBM Processing, IBM Corp., (Poughkeepsie, New York), 1965.

Leeds, H; Weinberg, G.; Computer Programming Fundamentals, McGraw-Hill, (New York), 1966.

Maram, G.; et al. "Cost Analysis - Primary Nursing," New England Deaconess Hospital, 1974.

McCracken, D.; A Guide to FORTRAN IV Programing, John Wiley & Sons, (New York), 1972.

Schlotfeldt, R.; "Reflections on Nursing Research," American Journal of Nursing, 60:492-494, April 1960.

Toward Quality in Nursing: Needs and Goals. Report of the Surgeon General Consultant Group on Nursing. U.S. Public Health Service publication #992, Government Pringting Office, (Washington, DC), February 1963.

APPENDIX 4

Hardware	Software
CPU	code
memory	instruction
core	mnemonic
word	location
bit	data
byte	
address	binary
K	octal
	floating point
	round-off
	truncation
	real
	integer
	BCD
	ASCII

Hardware (Cont.)
register
accumulator
contents
program counter
sign
processor
operator
operand
instruction
fetch
execute
clear
store
overflow

millisecond
μ second
nanosecond
cycle

input
output
peripheral
punched card
Hollerith
code
paper tape
mag tape
record
record gap
EOF
teletype
CRT
punch
console
sense switches

disc
floppy disc
track
sector
drum
auxiliary storage
random
sequential

parity
compatibility
I/O channel
interrupt
flag
access
read
write

A/D
clock
destructive
non-destructive
generation

Software (Cont.)
natural language
artificial language
semantics
syntax
vocabulary
parse
FORTRAN
COBOL
DATATEXT
BASIC

interpreter
compiler
assembler
machine language

absolute binary
relocatable binary
loader
linkage
subroutine
function
library

operating system
executive system

source
edit
program
module
path
flow
branch
transfer
control statement

variable
constant
parameter
argument

loop
sort

table
directory
pointer
procedure
algorithm
heuristic

index
dictionary
file
record
field
format
open
close

frame
test
debug
comments
reentrant code
housekeeping
initialization
structured program

Hardware & Software
off-line
on-line
transaction
real-time
batch
time-shared
interactive
data-base
MIS

deadlock
crash
partition
memory protect
recovery
rollback
journal

centralized
decentralized
distributed
minicomputer
microprocessor
configuration

queue
message
sattelite
concentrator
network

APPENDIX 5

Reference Readings for Students

Programming and Documenting Software Projects; P. J. Brown, ACM Computing Surveys, Vol. 6, #4, pp 213-220, December 1974.

Psychology and Program Design; David Frost, DATAMATION, Vol. 21, #5, pp 137-138, May 1975.

An Overview of Programming Practices; J. M. Yoke, ACM Computing Surveys, Vol. 6, #4, pp 221-245, December 1974.

Programming Style: Examples and Counter-examples; Kernigham & Plauger, ACM Computing Surveys, Vol. 6, #4, pp 303-319, December 1974.

Information Systems in Perspective; J. D. Aron, ACM Computer Surveys, Vol. 1, #4, pp 213-236, December 1969.

On-Line Management Information, Part I & II; N. J. Ream, DATAMATION, Vol. 10, #3, pp 27-30, March 1964; Vol. 10, #4, pp 39, 43, 45, & 50, April 1964.

Elements of Data Management Systems; George Dodd, ACM Computing Surveys, Vol. 1, #2, pp 117-133, June 1969.

Programming On-Line Systems I & II; Frank, Gardner, and Stock, DATAMATION, Vol. 9, #5, pp 29-34, May 1963; Vol 9, #6, pp 28-32, June 1963.

How to Prepare Functional Specifications; Joseph Rigo, DATAMATION, Vol. 20, #5, pp 78-80, May 1974.

HIPO for Developing Specifications; Martha Jones, DATAMATION, Vol. 22, #3, pp 112-114, 121, & 125, March 1976.

Flowcharting with the ANSI Standard: A Tutorial; Neal Chapin, ACM Computing Surveys, Vol. 2, #2, pp 119-146, June 1970.

Evolution of Business System Analysis Techniques; J. C. Couger, ACM Computing Surveys, Vol. 5, #3, pp 167-198, September 1973.

Sorting; William A. Martin, ACM Computing Surveys, Vol. 3, #4, pp 147-174, December 1971.

Supervisory and Monitor Systems, Robert Rosin, ACM Computing Surveys, Vol. 1, #1, pp 37-54, March 1969.

An Approach to Program Testing; J. C. Huang, ACM Computing Surveys, Vol. 7, #3, pp 113-128, September 1975.

Digital Control of Industrial Processes; Cecil L. Smith, ACM Computing Surveys, Vol. 2, #3, pp 211-241, September 1970.

Software Physics; Kenneth Kolence, DATAMATION, Vol. 21, #6, pp 48-51, June 1975.

Electronic Computers: A Historical Survey; Saul Rosen, ACM Computing Surveys, Vol. 1, #1, pp 7-37, March 1969.

Third Generation Computer Systems; Peter Denning, ACM Computing Surveys, Vol. 3, #4, pp 175-216, December 1971.

What Machines Can and Cannot Do; J. Nievergelt and J. C. Farrar, ACM Computing Surveys, Vol. 4, #2, pp 81-96, June 1972.

The Computer in the Humanities and Fine Arts; Sally Sedelow, ACM Computing Surveys, Vol. 2, #2, pp 89-110, June 1970.

Programming Time Versus Running Time; Lawrence Cooke, DATAMATION, Vol. 20, #12, pp 56-58, December 1974.

On-Line Versus Batch Costs; Edward Lias, DATAMATION, Vol. 20, #12, pp 69, 70, & 72, December 1974.

Terminals and Their Impact on Employee Motivation; Lowrie, Ryan, & Carlyle, DATAMATION, Vol. 20, #8, pp 59, 60, & 62, August 1974.

APPENDIX 6

Fundamentals of Computer Programming Preliminary to Nursing Research

1. Two number systems other than the decimal system are the
   1.
   2.

2. The octal system has ________ unique digits.

3. In the binary system all numbers are represented as combinations of ______ of two.

4. It is easy to convert binary numbers to octal and octal to binary because eight is equivalent to the third power of ____________.

5. A computer usually has the following parts:
   1.
   2.
   3.

6. Core memory is composed of ________ which are composed of individual ________.

7. Computers use the binary system because a bit can be electronically magnetized in two different ways: one way represents ________; the other represents ________.

8. Every letter of the alphabet, number, etc. must be coded in a ________ if it is to be stored in the computer.

9. A memory cell or word may contain a bit pattern which represents either ____________ or ____________.

10. Every computer is built to perform certain functions such as
   1.
   2.
   3.
   4.
   5.
   6.
   7.
   8.

11. A computer ________ is a series of ________ that tells the computer which functions to perform in order to carry out a specific job.

12. Instructions are usually stored ________ in successive words unless a specific instruction is to transfer to a location outside the ________ of the program.

13. The central processing unit contains components such as the ________ and ________ where all arithmetic types of operations are executed.

14. Two examples of sequential input devices are
   1.
   2.

15. Two examples of random input devices are
   1.
   2.

16. A random device is best for handling data that will be processed in ________ particular order because with a random device every piece of data is ________ accessible as any other piece.

17. Some input/output devices such as
    1.
    2.
    3.
    can also serve as ____________ devices.

18. An online or real time system implies

19. One of the standard input/output devices used in a real time system is a ____________________________.

20. In a real time system one drawback is the possible ____________ of input data due to ________________.

21. When programming real time systems you must write a special program called the __________________ or ____________ which __________________, the actual ____________ in which the various special logical functions of the system are carried out.

22. In a real-time or on-line system some functions are performed only when initiated by the arrival of appropriate ______________ to the computer. Other functions are performed with the elapse of certain ______________ intervals.

23. Since all data and instructions must be represented in binary format, programmers very often use a shorthand called ______________ to relieve themselves of the tedium of writing everyting with 0's and 1's.

24. The programmers are able to use this shorthand because it can be read into the computer as ____________ and another computer program called an ______________ will turn this data into the appropriate ________________ patterns.

25. There are higher level languages which permit the programmer to write his program in a language resembling scientific or business language. However, this type of input must be broken down by a computer program first to ______________ and then to ______________.

APPENDIX 7

Final Evaluation Date ______________

Please indicate the response to the objectives of this program:

Objective:

1. Code, score, and scale data from Nursing studies, "Orientation of the New Graduate at Peter Bent Brigham Hospital" and "Patient Attitudes Regarding the Functions of the Nurse in an Extended Role."

Objective Not Achieved	Achieved To Some Extent	Effectively Achieved

2. Utilize FORTRAN in computer programming.

Objective Not Achieved	Achieved To Some Extent	Effectively Achieved

3. Identify the basic components of a computer system and their functions.

Objective Not Achieved	Achieved To Some Extent	Effectively Achieved

4. Utilize a basic vocabulary of computer terminology.

Objective Not Achieved	Achieved To Some Extent	Effectively Achieved

Please indicate your overall reaction to this program series:

____Excellent ____Very Good ____ Good

____Fair ____Poor

GENERATIVE CAI FOR FOREIGN LANGUAGES

Henry W. Decker and Thomas Rice
University of California, Riverside

Presentation and correspondence: Henry W. Decker
Department of Literatures and Languages
University of California, Riverside 92521
(714) 787-3571, 787-5007

Introduction

Over the past several years we have developed a programming strategy for real-time generation of sentences of a foreign language, and for instructing the computer to perform grammatical operations upon such sentences. With this capability the computer is assigned a unique role in foreign language instruction: providing unlimited non-repeated examples of constructions and grammatical transformations that, typically, can be shown only once or twice in the textbook or on the blackboard.

We have experimented with techniques of program design using this capability in a variety of instructional formats, for drills, games, review, and testing, and have obtained positive student reaction through informal use of sample programs in the language laboratory and through integrated adjunct use in French for Reading Knowledge. At present we are seeking extramural funding for a couple of years of intensive software development, in order first to verify our strategy in adjunct programs for pilot sections of elementary French, German, Italian, Latin, and Spanish at UC Riverside, and then to turn our attention to its transportability.

Programming Strategy

The procedure for a given foreign language may be briefly summarized as follows.[1] We first construct a verb component which, given an infinitive, will compute any form of that verb. Input to the function will be values for the variables that specify verb forms in that language--usually the two parameters of person and tense, although Latin verb forms are four-dimensional (indicative/subjunctive and also active/passive as well). The function itself can be thought of as a set of rules for associating stems and endings, but of course details of morphological analysis will be language-specific; in addition, the analysis adopted will reflect whatever segmentation one wants to be able to recover when evaluating student responses, as Professor Weller showed in his paper on Spanish verbs presented at the 1976 CUCC.[2] External data available to the function must thus include at least a listing of any stems or forms that are irregular--that is, not predicted by the rules--and tables of endings. Output, in the model just sketched, will be a single verb form; alternatively, the verb component can be programmed to compute all forms of the verb it is given, in an output matrix.

Using such a component (which can be copied from one workspace into another) one can construct verb drills to suit any pedagogical purpose. A typical format shows one verb form as a cue with instructions to the student to type in some other form, such as the same person in a different tense or the subjunctive of the form shown. For the reading knowledge course we add a translating function to the verb component; the student identifies the tense of the verb form shown, and then is either given appropriate renderings in English or asked to supply one. In each workspace (as loaded from the library) we like to provide a variety of instructional programs with a menu listing them, so that students can switch programs freely without having to reload; for example, in a workspace labelled Italian verbs, there are choices of tenses, of verbs (-are, -ere, -ire, irregular), and of mixes for both. The computer selects verbs randomly within the appropriate category for any drill run and selects random values for variables not pre-set (for instance, random values for person where tense remains the same). In this way, consecutive runs are non-identical, but provide equivalent tests.

Where instructional programs require sentences we proceed to specify a predicate structure for each verb, roughly in the form: eat something, give something to someone, put something somewhere. The computer lexicalizes this frame by random selection of words within semantic and also pedagogical constraints: a word chosen as object of the verb eat must not only refer to something edible, but must also be a familiar vocabulary item at a certain level in a certain textbook. A noun-phrase component now applies morphological rules for agreement of adjectives, selection of noun-determiner, and the like, and sentence rules concatenate verb forms and noun-phrases in an appropriate order. Syntactic features left to random selection by the computer--such as whether the sentence is negative or interrogative or neither or both, or moving certain elements to the front of the sentence in German--contribute along with random vocabulary choices to produce a surprising variety of non-repeated sentences for, say, a drill on German *ein* words.

Sentences may then be passed through a transformational component including such functions as question formation, object pronoun substitution (*I gave my car to my friend*: *I gave it to him*), or relative clause embedding (*the car that I gave him*). Each component described above draws upon stored data (tables of endings and pronouns, lists of irregular forms, and coded idiosyncratic grammatical information), but it will be apparent that the component itself contains rules, generalized (algebraically) for all possible cases, such that the fixed set of components becomes a grammar of the language, expressed in some computing language. Instructional programs use this grammar to produce paired CUES and RESPONSES with which to elicit and evaluate student input (as shown here schematically):

```
CUE:      ICH HABE KEIN   HUND
STUDENT INPUT:          EN
EVALUATION:           (RIGHT)
RESPONSE: ICH HABE KEINEN HUND

CUE:      J'AI DONNE MON AUTO A MON AMI
STUDENT INPUT:        LE LUI
EVALUATION:           (WRONG)
RESPONSE:  JE LA LUI AI DONNEE
```

Grammar design must be effective in two respects: there must be sufficient randomization throughout to provide continuously varied output, but at the same time values for all grammatical parameters must be specifiable so that all output sentences will represent appropriate examples of a given grammatical problem. The grammar is obviously not generative in the usual sense of linguistics, where a grammar generates all the sentences of a language all the time; it is designed to crank them out, one by one, upon request. But it is not sufficient, either, simply to turn out sentences that are grammatically correct and semantically acceptable. For this grammar to be useful pedagogically, it must accept requests in the form, not "Compute a sentence of German," but "Compute a sentence of German such that *ein* appears in the masculine singular accusative form."

Language Learning and Generative CAI

In the Proceedings of the 1978 CCUC, Dr. Walter Maner observes that "the future of ordinary CAI, with its canned questions and repertoire of canned answers, grows dimmer by the day," and comments:

> It is not hard to see why. Once a CGAI program has been designed, it is capable of furnishing an inexhaustible supply of distinct problems (and solutions) for the student. By contrast, in a frame-oriented CAI system, the lesson designer typically programs each task as a separate case. Like ordinary programming but unlike conventional CAI, GCAI uses algorithms to manipulate structured data in order to produce output--in this case, a series of learning tasks for the student. There is no GCAI "language" in the tradition of GNOSIS, TUTOR, SCHOLAR-TEACH, COURSEWRITER, PILOT, and so on. Instead of constructing frames, the GCAI author uses some general-purpose, high-level programming language such as SIMULA, PL/1 or SAIL to build the data structures and algorithms which will (a) generate problems, (b) solve these problems, (c) present these problems (unsolved) to the student, (d) manage the dialogue between student and program (much as in conventional CAI), and (e) evaluate the student's problem-solving efforts (again, much as in conventional CAI).[3]

Language learning is a particularly appropriate field for the application of generative CAI. Practice *using* a new language (practice *saying things*) requires interaction with a human instructor, who can judge the effectiveness of the message. But what is being practiced --what is being learned when we learn a new language--is a new *code*, that is, a new vocabulary and in particular a new set of productive rules for constructing and combining forms. It is to the code-learning task that generative CAI is especially well-suited. Multiple-choice frames that have typified CAI have, for most fields of instruction, necessarily consisted of statements *about* some subject matter; in language instruction, however, sentences displayed by the computer are themselves *examples* of the subject matter--the well-formedness of those sentences *is* the subject matter. Since, furthermore, examples are generated in real time by the computer (rather than simply dealt out whole from stored banks of questions and answers), the rules that the computer applies are an analog of the rules that the student is learning, and it is in this special sense that the computer is equipped to present an unlimited number of problems to the student and evaluate the student's problem-solving efforts.

In one important respect our own experience does not bear out the characterization of generative CAI just quoted: we do not find use of a high-level programming language to be essential. As it happens, we began work on the project in APL; we are now working successfully in BASIC, time-sharing on a small-size computer (PRIME 400). This fact leads us to advocate a hands-on approach to computer programming on the part of foreign language instructors, an approach which seems to us in line with the current technological (and cost) revolution in computers. A healthy effect of that revolution has been to dispel the mystique of the big computers--that air of *for specialists only* that lasted well into the 1970s, and that in the field of educational software favored the production of packaged materials no more readily revised or reshaped by an instructor using them than printed textbooks are. Probably the first heralded--and, of course, still valid--claim for computer-aided instruction generally was its responsiveness to the student (immediate feedback, optional branching, etc.); attention should also be given to the responsiveness of software to the *teacher* using it--its pedagogical flexibility, as to items drilled, display format, manner of testing and scoring, order of presentation, and the like.

Professor Alfred Bork, at the 1978 Western Educational Computing Conference, described the production of educational software as still in the cottage-industry stage. In the Proceedings of that conference he points out the need for more effective methods of producing computer-based materials for large-scale distribution, in terms of a declining cost of software to match the declining cost of computers.[4] One can only agree. We are also arguing, though, that at least in the field with which we are familiar, *premature* emphasis on the packaging of materials for wide distribution could result in the language teaching profession's judging the possibilities of computer-aided instruction as such on the basis of only certain CAI materials--materials which may very well have missed some of the most intriguing of those possibilities for language learning, in particular, the special appropriateness and the programming flexibility inherent in *generative* CAI.

References

1. A more detailed description, in terms of specific programming problems and decisions, appears in Decker and Rice, "Computer-Aided Instruction in Foreign Languages," *Proceedings of the 1978 Western Educational Computing Conference*, pages 3-6. A report on initial research for the project appeared in Decker, "Computer-Aided Instruction in French Syntax," *Modern Language Journal*, LX, 5-6 (1976), pages 263-267.
2. Hubert P. Weller, "An Experiment in Computer Assisted Instruction in Elementary Spanish," *Proceedings of the Seventh Conference on Computers in the Undergraduate Curricula* (1976), pages 281-287.
3. Walter Maner, "Generative CAI in Aristotelian Logic," *Proceedings of the Ninth Conference on Computers in the Undergraduate Curricula* (1978), pages 117-126.
4. Alfred Bork, "The Declining Cost of Computers - Implications for Education," *Proceedings of the 1978 Western Educational Computing Conference*, pages 51-53.

MELODIC PERCEPTION DEVELOPMENT & MEASUREMENT THROUGH CAI

Douglas Ray and
Rosemary N. Killam
North Texas State
University, Denton,
Texas 76203

I. Introduction and Background

Musicians and non-musicians instinctively understand the importance of melody recognition. The ability to write melodies in musical notation has equally apparent value. However, students often have great difficulty in developing this latter skill.

The methods used to present materials for developing melodic perception and transcription skills have too often paid little attention to the measurement of the individual student's capabilities or problems. Students' melodic perception abilities must be accurately determined so that effective teaching and learning strategy can be implemented.

A computer-assisted instruction (CAI) system is used at North Texas State University to present a comprehensive program of ear-training lessons to undergraduate students. The system is described in detail by Hamilton and Scott, 1978. There are over 600 students, with widely divergent backgrounds and professional goals, enrolled in freshman and sophomore music theory classes at NTSU. These students must develop a variety of auditory skills to prepare for their prospective careers; the CAI system has proved valuable in student auditory skill acquisition.

Most previously-developed CAI curricula in ear-training have provided a group of short, simple melodies; such curricula prove too difficult for many beginning students. The authors have produced a package of lessons which presents melodic fragments abstracted from tonal subsets. Using these lessons, students can proceed through an ordered series of melodic constructs of increasing complexity. Analysis of data obtained from student use of this carefully structured curriculum will provide valuable insights into melodic perception and will enable further curriculum development to proceed from a logical basis. The influence of this research is not limited to current users of the system; the strategy of presentation can be adapted to improve teaching methods for the field of music as a whole.

II. Methods

A. CAI Hardware

The ear-training CAI system at NTSU employs the Automatic Music System (AMUS) designed by Dan W. Scott, Chairman of the Computer Sciences Department. The AMUS system consists of a Motorola 6800 microprocessor with a cathode ray tube (CRT) terminal and specialized music hardware (MUSOR). The microprocessor translates the musical score and accesses the MUSOR, which is the actual sound-producing apparatus engineered by Professor Scott. The AMUS system is connected to a Hewlett-Packard HP 2000-F Timeshared BASIC computer, which provides additional file storage capabilities. This system uses a flexible score language for notation input and offers a variety of timbre and articulation possibilities. Of primary importance is the facility of rapid playback which is a substantial advantage over other computer-based music systems in this cost range.

B. CAI Software and Lesson Design

The curriculum driver and system operation programs (designed by Richard Hamilton) provide optimum effectiveness in lesson presentation.

A system of graphic notation was developed in the summer of 1978 (Hamilton and Killam, 1979). The student, after hearing a melody presented by the CAI system, places the first note at the proper location on the staff by moving the cursor up and down, then typing an abbreviation for the note's duration (for example, Q for quarter note). When the student presses the RETURN key, the computer prints the note symbol for the value which the student typed. The student moves the cursor to the next pitch position, and the process is repeated. At the end of the melodic example, the student types "//", which indicates that his or her answer is ready for evaluation. A correct answer receives the reply, "GOOD. THAT'S RIGHT." If the response is incorrect, the staff is redrawn showing only that part of the student's answer which was correct, and the program displays the message, "SORRY. TRY AGAIN ?". The student is allowed a specific number of trials, after which the right answer is displayed on the screen, the music is replayed, and a new melody is presented. The student may call for repetition of the music at any time by typing a question mark. There is no limitation on the number of times the music for a particular example may be played.

Before each melodic pattern is played, the tonic triad is sounded to establish key feeling. This is followed by a tempo indicator, which is the tonic note repeated for the correct number of beats per measure at the interval of an octave above the rest of the example.

The lesson continues to present examples selected at random from the lesson file until the student completes a specified number of melodies on the first trial. The program then asks, "WOULD YOU LIKE TO TAKE ANOTHER LESSON?". The student may continue by typing YES or may terminate the session by typing "//STOP". The "//STOP" feature is also available at any time during the lesson.

C. Student CAI Use

Students are advised to take an introductory lesson which focuses on basic computer interaction. The use of the CTRL key in combination with H for backspacing is unfamiliar to anyone not previously acquainted with the HP system. The CTRL X feature, which cancels an entire line, is also introduced. After some practice with these features, a very simple example demonstrates the graphic capabilities used in the melodic dictation exercises, beginning with a one-note example which the student enters and has the computer check. The text of this example is:

```
MOVE THE CURSOR UP TO THE "C" ON THE
THIRD SPACE.

TYPE "Q" FOR QUARTER NOTE.

TYPE THE RETURN KEY AND I'LL DRAW
YOUR NOTE.

NOW TYPE TWO SLASHES "//" AND I'LL
CHECK YOUR ANSWER.
```

The most effective way to avoid user frustration and to develop student entry skills necessary for melodic dictation is to begin with exercises which require note entry only. The student must have a solid grasp of the answer format, so that he or she may concentrate on the music as the complexities of the melodies increase.

Each lesson provides a restatement of answer format which the user may elect to skip if he or she is comfortable with the note entry procedures. Each lesson contains a brief summary of pertinent information to refresh the memory of those who need it.

D. Lesson Content

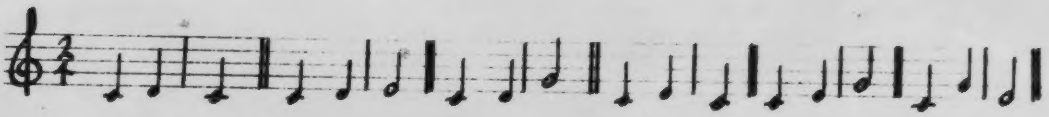
Since the majority of popular music and a large portion of the serious music accessible to today's listener is based on the principles of tonality inherent in most music written since the 1600s, tonality is the most logical musical style with which to begin ear-training. Students' continual contact with tonal music provides a familiar basis, and thus an easier starting point for formal ear-training. Knowledge of the tonal idiom is essential to students' expansion into contemporary and modal music.

The lessons present short melodic patterns of three, and then four, notes which use simple rhythmic durations of at least a quarter note. The simplification of rhythm allows students to focus on the pitch. All examples are strongly rooted in the tonal tradition. Each fragment either begins or ends with the pitch C in the keys of C major or C minor. The key limitation controls for variance of students' pitch perception.

Each example is selected at random from a large number of possibilities stored in the lesson file. Random presentation provides subsequent data analysis with controls for learning effects; the difficulty of individual patterns can be more easily ascertained if they are not presented in the same order. Thus, the analysis of melodic perception is aided by specifying limitations on frequency range, interval range, rhythmic simplicity, and random presentation.

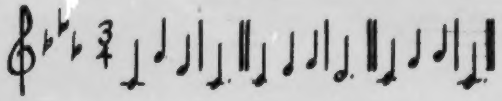
Lesson content was determined by preparing all possible combinations of three and four notes within a pitch range of C4 up to G4 and C5 down to G4. Examples of these patterns are shown in Example 1.

The placement of the tonic note (C) at the beginning or end of each fragment is consistent within a lesson, but varies from one lesson to the next. Additional restrictions are imposed by certain tonal relationships standard to the Common Practice period. In cases where the fourth scale degree functions as the seventh of the dominant chord, the note following the fourth resolves down to the third. When the leading tone has a strong cadential feeling, it resolves upward to tonic on the next note. The use of these techniques provides musical examples with melodic procedures familiar to the student, and helps him or her gain maximum learning from the lessons.



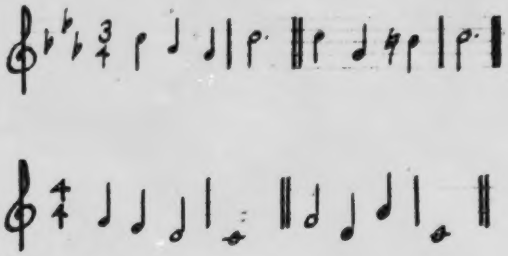
Example 1.

The following melodic fragments are examples of four-note patterns presented in the lessons.



Example 2.

Fragments using a range of G4 up to C5 required more editing, particularly those in minor keys. Ascending and descending scale fragments were screened to avoid the augmented second between the lowered sixth and raised seventh scale degrees. In addition, three different patterns of rhythmic presentation are used, as illustrated in Example 3.



Example 3.

III. Results

The range limitation imposed on the fragments changes as the student proceeds through a series of lessons. Lessons 1 through 5 use only three-note fragments. The first lesson presents patterns in C major with intervals taken only from within the interval of the fifth--C4 up to G4. The unique relationship of tonic to dominant and the frequency with which it is encountered in music literature led to its use as a delimiting factor. Lesson 1 always presents the tonic first. Lesson 2 always ends with tonic. Lessons 3 and 4 offer the same material in the minor mode.

In Lesson 5 the student is introduced to the intervallic possibilities in the tetrachord from C5 down to G4. Both major and minor keys are used in this lesson.

Lesson 6 introduces four-note fragments in C major, using intervals encompassed by the scale tones of C4 up to G4. Lesson 7 uses the same format, except that all examples end rather than begin on tonic. Lessons 8 through 12 are the culmination of this learning sequence and combine intervals from the fifth below the tonic to the fifth above, in any order, using both major and minor keys. The number of possibilities for four-note patterns is so great that four lessons were used to present these fragments.

Table 1 summarizes the melody lessons:

Table 1.

Lesson Title	Pattern	Notes Used
M1	major--three notes starting with tonic	C up to G
M2	major--three notes ending with tonic	C up to G
M3	minor--three notes starting with tonic	C up to G
M4	minor--three notes ending with tonic	C up to G
M5	major & minor--three notes starting or ending with tonic	C down to G
M6	major--four notes starting with tonic	C up to G
M7	minor--four notes starting with tonic	C up to G
M8	minor--four notes ending with tonic	G up to G
M9-12	major--four notes ending with tonic	G up to G

IV. Discussion

Melody is a fundamental component of music; the perception of melodies is of prime importance in auditory training. Melodic dictation requires the student to listen to a group of notes and then transcribe them into musical notation. Melodic dictation has two primary goals for the student: 1. the identification of patterns of melodic motion; and 2. the ability to translate auditory information into a precise symbolic notation. Each of these aspects has obvious prerequisites, such as:

1. The student must have the ability to recognize intervals (the relationship between two notes) at a fairly rapid rate. Interval recognition leads logically to study of three and four notes in succession, followed by melodic fragments of gradually increasing length.
2. The student must be familiar with rhythm and rhythmic notation. The duration of each note must also be transcribed into musical notation. With competence in both of these skills, students can begin to transcribe melodies accurately.

CAI offers the following advantages to to melodic perception research:

1. A vast quantity of curriculum materials can be made readily available.
2. Lessons can be structured so that each promotes a specific goal with melodies of equal difficulty.
3. The lesson programming allows examples to be chosen at random from a number of possibilities.
4. The presentation itself, once systematized, is always consistent.

5. The possibility of teacher bias toward a particular curriculum area, method of presentation, or individual student is eliminated.
6. The CAI system provides for accurate evaluation of student data.

Record-keeping facilities maximize curriculum effectiveness through storage of detailed student data, which are subjected to sophisticated on-line analysis for each student, question, and type of question. The results of this study can be used for future development and implementation of improvements in teaching strategy. Analysis of the data obtained from student use of the package will provide new insights into the whole process of melodic perception.

The focus on pattern perception allowed by the control of other musical factors--rhythm, frequency range, and duration--will facilitate a thorough and accurate analysis of student data. There is no other effective way to present such a quantity of materials in so consistent a fashion. Since these lessons are centered on skills improvement, extensive and ordered curriculum is required. With a total of 600 freshman and sophomore students who may use these lessons, information will accumulate quickly. Constant re-evaluation of student melodic perception skills will aid in lesson format and learning strategy improvement. The use of these lessons during the Spring 1979 semester will provide in-depth information on student learning processes. These data will be reported at the NECC Conference.

V. Summary and Prospects

In conclusion, the effectiveness of any educational strategy is limited by its appropriateness both to the material presented and the people for whom it is intended. An effective presentation plan must be sensitive to student response and learning. Research into and development of improved strategies in melodic recognition and pattern perception provides valuable information to the field of music as a whole. CAI has proved a most effective vehicle for such research. The series of lessons described in this paper are a valuable addition to the research effort; the carefully controlled melodic content will facilitate data analysis.

Many possibilities remain to be explored within the framework of the lessons described above. Transposed versions of each of these lessons are planned for future implementation. The authors also plan to provide each student with a summary of his or her achievements at the conclusion of a lesson, which will include: 1. acknowledgement of success; 2. report on number of trials used for correct answers; 3. identification of interval perception difficulties combined with appropriate review suggestions.

There is much more to be studied with regard to the effects of rhythm, frequency range, and interval size, before a clearer view of sound pattern perception is within reach. The melodic lessons developed for this study are the beginning of what is hoped will be a more rigorous approach to skill development and its relation to research. The benefits of increased interaction in these two areas are enormous.

Selected References:

Hamilton, Richard L. and Dan W. Scott. "A New Approach to Computer Assisted Instruction in Music Theory", Proceedings of the Ninth Conference on Computers in the Undergraduate Curriculum, Denver, 1978.

Hamilton, Richard L. and Rosemary N. Killam. "Cost-Effective Implementation of Music Notation on CRT Terminals", Proceedings of the 1979 Association for Development of Computer Instructional Systems Conference, San Diego, 1979.

Killam, Rosemary N. and Paul V. Lorton. "Recognition of Tonal Sequences Derived from Traditional Triad Structures", presented at the 94th Meeting, Acoustical Society of America, Miami, 1977.

APPENDIX

SAMPLE STUDENT INTERACTION

```
WHICH LESSON WOULD YOU LIKE TO TAKE? M1

 DOUG, HERE IS SOME MELODIC DICTATION--

 I WILL PLAY THREE-NOTE MELODIES FOR YOU.

 YOU ENTER B O T H THE MELODY AND THE RHYTHM.

 HERE'S HOW IT WORKS---

 THE  C U R S O R  IS THE WHITE SQUARE MOVING ON THIS SCREEN.

  MOVE THE CURSOR TO THE LINE OR SPACE FOR THE PITCH YOU WANT.

   --TO MOVE IT UP:  HOLD DOWN THE "CONTROL" KEY & TYPE  " K "

   --TO MOVE IT DOWN: HOLD DOWN THE "CTRL" KEY & TYPE " J "
       (J & K HAVE ARROWS ON THEM TO SHOW THE DIRECTION)


 DOUG, AFTER YOU MOVE THE CURSOR TO THE STAFF PITCH YOU HEARD--

     (1) TYPE THE LENGTH OF THE NOTE

     (2) THEN TYPE THE "RETURN" KEY

     (3) THEN TYPE 2 SLASHES ( // = DOUBLE BAR ) & RETURN

 DOUG, LET'S TRY IT OUT, JUST FOR PRACTICE.

   (1) MOVE THE CURSOR UP TO THE " C " ON THE 3RD SPACE

   (2) TYPE  Q  FOR QUARTER NOTE

   (3) TYPE "RETURN"-- & I'LL PRINT YOUR NOTE!!

   (4) TYPE 2 SLASHES ( // = DOUBLE BAR ) & RETURN

   (USE THE  /  ON THE QUESTION MARK KEY.)
```

DOUG, USE THESE LETTERS FOR NOTE LENGTHS--

W = 𝅝   RW = 𝄻
H = 𝅗𝅥   RH = 𝄼
Q = ♩   RQ = 𝄽
E = ♪   RE = 𝄾

ADD A . (PERIOD) TO NOTES, FOR DOTTED NOTES

%S %E

DOUG, HERE'S AN EXAMPLE, WITH ANSWER--
2
4

REMEMBER--

(1) HOLD DOWN THE CONTROL KEY & TYPE J--TO MOVE CURSOR DOWN

(2) HOLD DOWN THE CONTROL KEY & TYPE K--TO MOVE CURSOR UP

(3) TYPE THE NOTE LENGTH - W H Q E (OR- RW RH RQ RE FOR RESTS)

(4) TYPE "RETURN" AFTER EACH NOTE & " // " AFTER ENTIRE EXAMPLE

(5) TYPE " ? " TO REHEAR THE MUSIC
THEN ENTER THE NOTE VALUE - I'LL PUT IT AT THE RIGHT PLACE.

(6) USE ONE SLASH / FOR BAR LINES.

(7) IF YOU SHOULD MISS ON YOUR FIRST TRIAL, I'LL RE-DRAW THE
PART YOU GET RIGHT. THEN YOU FILL IN THE REST CORRECTLY.

(8) TYPE "//R" IF YOU WANT ME TO CLEAN OFF ANY TYPO ERRORS.

(9) TYPE " //HINT " TO SEE THE INSTRUCTIONS AGAIN.

DON'T FORGET BAR LINES (TYPE ONE / ONLY).

I'LL BLIP THE TEMPO WITH TONIC NOTE AN OCTAVE HIGHER.
THIS IS NOT PART OF THE EXAMPLE!

PLEASE DON'T USE "CTRL" WITH L - I CAN'T COPE WITH IT.

THE FIRST NOTE IS ALWAYS TONIC AND ALL EXAMPLES ARE IN MAJOR.

THE CURSOR WILL START AT MIDDLE C - DON'T GO ANY LOWER!

LET'S GET STARTED, DOUG!

THE STANFORD LOGIC COURSE:
DESIGN, DISSEMINATION, AND DEMONSTRATION

Tryg A. Ager
Department of Philosophy
State University of New York at Binghamton

James McDonald
Institute for Mathematical Studies in the Social Sciences
Stanford University

Abstract

After more than a decade of development, the Stanford computer-based course in elementary symbolic logic is one of the largest and most complex CAI courses in the world. The purpose of this paper is to discuss the current design philosophy of the course, both in relation to the real constraints imposed by the fact that it is heavily used at Stanford, and in relation to the pedagogical and mathematical ideals the course tries to realize. Additionally we will discuss the strategy of a current venture by Stanford and SUNY at Binghamton to transport the logic course. At the conference we will provide an on-line demonstration of the course.

1 Introduction

The Stanford logic program, called VALID (Voice-Assisted Logic Instruction Dialogue), is currently and since 1973 has been the only way introductory symbolic logic is taught at Stanford.* VALID is mainline CAI with the primary instructional burden placed on VALID itself. This distinguishes it from most other CAI applications in symbolic logic. Whereas BERTIE [Moor and Nelson, 1976], DERIVATIONS [Ager, 1976], and EMIL [Garson, unpubl.] are tutorials designed to supplement a regular lecture course, VALID supplants the lecture course. VALID differs from ENIGMA [Laymon and Lloyd, 1976], the only other mainline logic program in regular use, in its subject matter. ENIGMA is mainly instruction in techniques of recognizing and evaluating informal inferences. VALID concentrates on the exploration of the operation and significance of a logical calculus. VALID is a large program. Including the curriculum files, VALID sources occupy about 2 million bytes. It requires an auxiliary computer to produce the synthesized speech constituting the audio portion of the program. It is computationally complex in its parsing, proof-checking, and control structures.

Students require about 70 hours of connect time to complete the core of the course and up to 15 more connect hours to finish the requirements for an A grade. Another 20 hours may be spent away from the terminal working on finding axioms exercises and the optional branches. Enrollment varies between 50 and 90 students a quarter.

* The program has an evolutionary history documented in Suppes, Smith, and Beard, 1977. This discussion of VALID owes much to the efforts of previous workers on the project, and especially to Prof. Patrick Suppes under whose general direction projects in computer-assisted instruction in logic at The Institute for Mathematical Studies in the Social Sciences have continued for nearly two decades. Robert L. Smith, now at Rutgers, was a principal architect of VALID. Lee Blaine is the designer of the proof reductions. Lawrence Markosian, Michael Hinckley, Mark Davis, and Robert Ladaqqa worked on the programs and curriculum of VALID. James McDonald and Jos Marlowe are working on its transportability. The development of VALID is supported by the National Science Foundation, grant number SED-77-09698. The work on transportability of VALID is supported by the Fund for the Improvement of Postsecondary Education.

We will describe the program in two ways, giving first a student's view of what the program does and how it behaves. Later we will turn to the designer's perspective and go over more technical and theoretical aspects of VALID.

2 VALID: The Student Point of View

Students who enroll in VALID are given a reference manual and shown to the student terminals. From then they are on their own, except for consulation with teaching assistants and peer interactions. The learning is self-paced and self-motivated. Since the student point of view is not insensitive to the matter of grades, we mention that at Stanford VALID grades are based on how much of the course has been completed. It requires about 70 hours of connect time plus 10 to 15 hours off-line to complete the core or pass portion of the course. Completion of the additional segments required for an A grade may take another 20 hours of connect time and some additional homework hours. No tests are given, but they could be. Mandatory but staggered milestones are set at the beginning of the term both to smooth out peaks in the computer usage curve and to encourage reasonable student work schedules. Students may exceed the milestones, but are not permitted to fall behind them. Mandatory milestones have been observed to increase the pass rate in the course by about 5% so that now about 75% of the enrollees end with pass or better as a grade. Since Stanford does not record a failing grade, it is incorrect to infer that the other 25% of the students have failed VALID. Some students have simply dropped the course. Although Stanford does not use off-line testing in connection with VALID, there is no reason why implementations of VALID at other colleges and universities should not use examinations, off-line or on-line.

2.1 Course Content

A student taking VALID will encounter the following sequence of topics in elementary symbolic logic.

1) The informal and formal (truth functional) meaning of logical concepts such as and, or, if-then, not, and if and only if.

2) Construction of derivations in propositional logic.

3) Formal counterexamples and methods of refutation of inferences in propositional logic.

4) An introduction to axiomatic systems using integer arithmetic as well as the theories of commutative and non-commutative groups.

5) Finding Axioms: From a list of formulas find a subset from which the rest can be derived.

6) Symbolizing English sentences in first-order logic.

7) Derivations and counterexamples in first-order logic.

8) Consistency of premisses and independence of axioms.

The inclusion of the lessons on integer arithmetic illustrates the relationship between logical and mathematical subject matters, and is an important addition to the standard elementary logic topics.

In addition to the core topics above, VALID has three branches which are taken for B or A grades. First there is social decision theory in which the logic of preference is studied and Kenneth Arrow's theorem that under certain plausible conditions rational methods of social choice are impossible is proved. Second, there is a branch giving a general introduction to Boolean algebra in which techniques for visual proof methods using diagrams are being developed. Third, there is a branch on qualitative probability in which a non-quantitative foundation for probability is developed following Suppes, 1973. A branch on elementary metamathematics exists and awaits implementation.

The core topics plus the branches present students with a rigorous course in elementary logic, axiomatic theories, and applications of logic.

2.2 Lesson Types

We will now outline some of the lesson types in VALID. Since students are not required to use a textbook, new concepts are presented using test and branch lesson frames which require only short or multiple choice responses. But these question and answer frames are mainly expository and transitional in purpose. Derivation construction is the heart of VALID.

A derivation is a stepwise demonstration that a conclusion follows from a set of premisses. In its straightforward uses, the proof checker permits step by step construction of derivations with instant feedback and error diagnosis. Students are gradually taught an extended rule set for natural deduction which includes the basic introduction and elimination rules for each connective and common derived rules such as modus tollens, disjunctive syllogism, and DeMorgan's laws relating conjunction, disjunction, and negation. Each rule of inference is embodied as a command to the computer. The command 4,8 AA means that the computer should detach the consequent of an implication found either on previous line 4 or 8 on the grounds that the other line is the antecedent of the implication.

Here is a simple derivation in propositional logic using the proof checker. The student input is underlined, a convention followed throughout this paper. The star is a prompt character indicating that some input is required.

```
Derive:  R & Q -> NOT(NOT R OR NOT Q)

*HYP
HYP        (1)  R & Q
*WP   (2) *NOT R OR NOT Q
*1 SIMP
1 SIMP 1 (3)   R
*3,2 DD
3,2 DD     (4)  NOT Q
*1 SIMP 2
1 SIMP 2 (5)   Q
*2,4,5 IP
2,4,5 IP (6)   NOT(NOT R OR NOT Q)
*1,6 CP
1,6 CP     (7)  R & Q -> NOT(NOT R OR NOT Q)

CORRECT
```

The strategy of this derivation is to assume the left side of the conditional to be derived and show that the right side follows. Thus the appropriate working premiss is selected by the command HYP. On line 2 another assumption is made, this time for the purposes of a reductio ad absurdum argument. The student had to type in the whole formula. But notice that it is the only student-entered formula in the whole derivation. VALID selects or constructs the rest upon receipt of commands. The command 1 SIMP means that the first conjunct of line 1 is desired as line 3. The command 3,2, DD requests that the result of applying the disjunctive syllogism rule to lines 2 and 3 be taken as line 4. Strictly, NOT NOT R is required as the denial of NOT R, but VALID operates modulo double negation in the interest of suppressing trivial details. To get line 5 the right (or second) conjunct of line 1 is selected by the command 1 SIMP 2. The SIMP rule has the form i SIMP j which means select the jth conjunct of the conjunction on line i. Now lines 4 and 5 contradict, so line 2, from which the contradiction followed, must be false. That is the rule of indirect proof or reductio ad absurdum. Finally, since now the right side of the conditional being derived has been deduced from its left side, the rule of conditional proof can be applied for the final line.

Bookkeeping details about which lines logically depend on which others are kept internally by VALID, but may be displayed at student option, and are displayed during the first few lessons in which logical dependence is discussed.

Here is another derivation which illustrates how VALID handles derivations involving quantifiers. The point to be proved--that if there is someone who loves everyone then everyone is loved by someone--reflects an important fact about order of quantifiers in formal logic. The converse of this relationship does not hold, since in matters of love, each to his own does not imply each to the same. This derivation illustrates the introduction and elimination rules for each quantifier.

```
Derive:  (A y)(E x) S(x,y)

P          (1)  (E x)(A y) S(x,y)
*1ES
Variables:  $  x
Specify for x:  $  x
1ES        (2)  (A y) S(x,y)
*2US
Variables:  $  y
Substitute for y: $  y
2US        (3)  S(x,y)
*3EG
Replace term: x
Variable: x
3EG        (4)  (E x) S(x,y)
*4UG
Generalize: y
4UG        (5)  (A y)(E x) S(x,y)

CORRECT
```

The flagging, subscripting, and dependence information in terms of which the restrictions on the quantifier rules are stated are all handled internally. When students first learn the quantification rules, however, all information is displayed. In particular, ambiguous names instead of free variables are used for existential specifications.

The quantification commands are easy to use. At any input request within a quantification rule there is a default which, though basically a guess by the computer, turns out to be just what the student wants in most cases. Thus at line 2 the program defaults to x as the term of specification. The student requested the default by just typing escape ($). At line 3 an escape in response to the request "Variables:" results in all initially universally quantified variables being specified. If (A x,y,z) S(x,y,z) were the formula, then an escape in response to "Variables:" would result in x, y, and z being specified. In this way more than one variable can be specified in one step. Likewise, more than one term can be generalized in a single step.

VALID also automatically reletters formulas to avoid unintended variable capturing. From

```
(A x) S(x,y)
```

students can use existential generalization to replace the term y by the variable x, but the result will be

```
(E x)(A x1) S(x1,x)
```

not the incorrect

```
(E x)(A x) S(x,x)
```

where the second argument of S has been illegitimately captured by the universal quantification of x.

Like the propositional rules, the quantification rules are designed so that as students gain familiarity with VALID and competence in logic they can use more and more of the powers built into the rules. As elsewhere in VALID, as students learn more, VALID is able to challenge them more.

Another example of how VALID makes more powerful rules of inference available is the rule of tautological implication. Once students have learned the basic rules for derivations in propositional logic, they are given the tautological implication rule. With this rule they can enter a formula, claim that it is a tautological consequence of some preceding steps in the derivation, and then have the tautological implication rule confirm it. Or if their claim is incorrect and it is not a tautological consequence, VALID will refute it with an explicit counterexample.

In addition to increasingly difficult derivation problems for which increasingly sophisticated rules of inference become available, the VALID student encounters lessons where the proof checker is capable of accepting axioms or previously established theorems as lines in a derivation. These features are used in the integer arithmetic section where axiomatic theories are studied and in the the finding axioms exercises.

3 The Designer's Point of View

One of the most prominent design objectives of VALID is to build a proof checking program that is completely rigorous on the one hand but permits the suppression of trivial detail. The result is a standard of informal rigor which is intended to approximate the kind of reasoning found in standard mathematical and logical expositions. Accordingly, VALID does not exploit the computer environment to promote a mere symbol pushing approach to logic. Instead, students are encouraged to make greater and greater conceptual jumps in consecutive lines of a derivation. We mention now two ways in which compression of trivial details and reasoning in large pieces are permitted in VALID.

First, there is a provision in VALID for nesting a series of commands. Suppose one sees that the result of a use of modus ponens on two lines i and j can be used with yet another line k and the disjunctive syllogism rule to derive a desired formula. The command

```
(i j AA) k DD
```

does these operations in one step. So if students can see ahead several steps they can issue complex commands to the checker and thereby shorten proofs.

Second, in the study of axiomatic systems it is frequently necessary to apply an axiom or a previously proved theorem to a given line. Since theorems and axioms are often universally quantified they need to be specified for the case at hand. The special rule of inference, IMPLIES, will apply a previous axiom or theorem to the present case, and infer the appropriate conclusion. We give a schematic example where the axiom to be applied will be (A x,y)(B(y,x) -> C(x,y)), and be known as Axiom 1. We enter the derivation at line i where B(v,w) has already been established

```
[Given]          (i) B(v,w)
*i AXIOM 1 IMPLIES
i, AX 1 Implies (j) C(w,v)
```

The IMPLIES rule is the computer-based equivalent of the common practice of justifying a step of logical or mathematical reasoning by saying, "Using Axiom 1 we get ..." Such a rule of inference permits students to develop their logical intuitions about which previously established facts apply to present cases.

3.1 Proof Reductions

A fancier piece of proof discovery machinery currently being implemented has to do with the reduction of a complex proof to simpler and simpler subproofs. Given a formula to be derived from a set of premisses, this machinery will permit students to develop an analysis of the problem by a series of reductions of the problem. Suppose that the problem is to derive the theorem of the transitivity of implication. The theorem itself is the main goal:

```
GOAL 1: (P -> Q) & (Q -> R) -> P -> R
```

To use proof reductions, the command REDUCE is entered and a reduction strategy is chosen. In this case the appropriate reduction is CONDITIONAL. The problem now reduces to deriving a subgoal from an assumption.

```
GOAL 2:  P -> R
assuming (1) (P -> Q) & (Q -> R)
```

Since the new goal is again a conditional, this time P -> R, using CONDITIONAL reduces the problem to

```
GOAL 3:  R
assuming (1) (P -> Q) & (Q -> R)
assuming (2) P
```

No further reduction of goals can occur. P must now be derived from the two assumptions.

```
1 SIMP    (3) P -> Q
2,3 AA    (4) Q
1 SIMP 2  (5) Q -> R
4,5 AA    (6) R
```

R, the current goal, has been reached. Because the goals have been introduced in such a way that reaching every goal subordinate to a given goal entails reaching the given goal, the problem now can be unwound recursively with the command DONE. The unwinding gives two new lines which complete the proof of the transitivity of implication. Both lines 7 and 8 are generated by the DONE command.

```
DONE
2,6 CP    (7) P -> R
1,7 CP    (8) (P -> O) & (O -> R)  -> P -> R
```

This is a simple example of proof reduction. The method accomodates branching goal trees such as are found in proofs by cases and has been extended to include the defined relations in set theory.

Let us apply the strategy-based approach of proof reduction to the problem about quantifiers we used as an earlier illustration. The problem was to derive

```
GOAL 1:  (E x)(A y) S(x,y) -> (A y)(E x) S(x,y)
```

which is the top-level goal. Again the goal is a conditional so CONDITIONAL reduces the problem to

```
GOAL 2:  (A y)(E x) S(x,y)
assuming (1) (E x)(A y) S(x,y)
```

Goal 2 can be reduced by UNIVERSAL, giving

```
GOAL 3:  (E x) S(x,y)
assuming (1)  (E x)(A y) S(x,y)
```

as the current problem. Goal 3 reduces by EXISTENTIAL to

```
GOAL 4:  S(x,y)
assuming (1)  (E x)(A y) S(x,y)
```

Now it is time to start working on the assumption that has been carried along. Successive applications of the rules of existential and universal specification give the additional lines

```
1 ES      (2) (A y) S(x,y)
2 US      (3) S(x,y)
```

The current goal has been reached. It is time to climb back down the goal tree with the command DONE, to complete the proof.

```
DONE
3 EG      (4) (E x) S(x,y)
4 UG      (5) (A y)(E x) S(x,y)
1,5 CP    (6) (E x)(A y) S(x,y) -> (A y)(E x) S(x,y)
```

The goals were introduced so that completion of the subsequent goals entail completion of previous ones. Thus reaching goal 4 on line 3 means that we can get goal 3, which yields goal 2 on line 5 and finally we reach the main goal on line 6.

The goal reduction machinery is pedagogically significant because it directly implements strategy-based problem solving methods. Students are encouraged to think in terms of global strategies. This contrasts strongly with pattern-matching approaches to proof discovery in formal logic. We believe that not only are strategy-based methods more readily extendable to problem solving skills in domains other than logic, but also they are more directly applicable to reasoning as it actually is done in higher mathematics. The solution of a problem by reduction mechanisms corresponds more closely to intuitive procedures used by logicians in analyzing and approximating solutions to complex problems.

The proof reduction system can be automated, giving an elegant and highly intuitive theorem proving program. Or it can be implemented in such a way that it can dynamically provide guidance to students having difficulty in constructing a proof.

The reduction method also permits the issuance of promissory notes on lemmas as long as the lemmas are related to steps in the existing reduction chain. A student might see that <u>if</u> a certain relationship holds then the problem could be solved. The reduction mechanism will immediately verify the hunch that the relationship fits into the reduction chain and will hold the mortgage on the lemma until the student supplies a proof for it as well. When all such promissory notes are paid off the problem is solved.

The purpose of including these sophisticated inference procedures in the logic course is to permit maximum development of students logical intuitions. There are, especially in the optional branches of the course, formal inferences which get very messy if attempted using only the primitive rules of logical inference. Derived rules, nested commands, the implies rule, and the proof reduction rules provide more and more powerful tools for problem solving which minimize trivial detail and focus students attention on essential elements of reasoning.

It is easy to restrict students in a computer-based instructional setting, but VALID resists restriction and tries to develop the students powers of logical intuition as well as their competence in manipulating formulas according to the basic machinery of a logical calculus.

4 Dissemination of VALID

Because of widespread interest in VALID a project to make it transportable is currently underway. We will briefly describe the transportation strategy here. A fuller discussion can be found in Ager and McDonald, 1978.

The size and complexity of VALID make the normal methods of transport unusable. Because of their highly recursive structure, VALID program modules are not easily written in the usual languages of transport--FORTRAN, BASIC, and COBOL. LISP, however, is quite compatible with the kinds of functions found in VALID and because much work has been done on LISP portability [Hearn and Griss, 1977], we chose LISP as our transport language.

Portable LISP, called Standard LISP by Hearn, comes with an interpreter, a compiler, and a source program such as VALID or REDUCE such that the whole system can be bootstrapped on most IBM 360 and 370 and DEC 10 and 20 series computers. Our goal is to have an identical source which then can be implemented on different computers using different interactive monitors. Machine dependencies are introduced only at the level of the interactive I/O and Standard LISP interpreter. Thus it will be possible to distinguish source code maintenance, which will be machine independent, from interactive I/O and interpreter maintenance which will be machine and operating system dependent. We will have converter schools representative of the major machines and interactive systems manage the latter problems while IMSSS manages the VALID source code.

Current plans call for implementations on DEC 10 TENEX and TOPS-10 systems as well as IBM 370 with CMS and the MENTEXT variant of the WYLBUR-ORVYL-MILTON system by the summer of 1980. Because of the requirement of an auxiliary computer to handle the speech synthesis, the transportable VALID will be mute. Nevertheless it will be designed to use the limited graphics features of most modern display terminals. Thus to the student the terminal screen will be more like a blackboard where text with diagrams can be displayed and modified and less like ancient scrolls of sequential text.

To make portable VALID flexible, authoring capabilities will be transported along with the course curriculum as it exists at Stanford. Logic instructors, even though they are not programmers, will be able to write their own logic lessons, make changes in the rules used in derivations, rearrange the lessons, and adapt the program to varying environments and purposes. Because distributed software usually gets modified, we expect to see interesting new uses for VALID once it is transported. **

The economics of transportability includes two factors. First, repeated development costs can be avoided as a program becomes portable. Instructors can use the programs for teaching logic right away at minimal costs. Second, there is the cost of computational resources for running VALID. This is the question most often raised about portability and the least easily answered. The reason for the difficulty is that the same unit of computation has different costs at different university computer centers.

At this point we can affirm that development costs of programs as large and complex as VALID cannot be borne by most colleges and universities. So on that point, transportability makes economic sense. But because local charging algorithms and allocation policies differ greatly, we will not venture cost projections for the typical

---

** Instructors interested in the possibility of using VALID should contact either author.

VALID user. We know, however, that in the fall of 1978 average cpu use at Stanford was about two minutes per hour of connect time for VALID students.

VALID represents a new level of sophistication in widely available college-level instructional computing, and the numerical costs must be balanced against the educational benefits and instructional opportunities it presents.

One approach we will be trying is regional networking of the program. At least two propspective users are network centers. As to the possibilities of running VALID on a personal computer, no current micro has either memory or mass storage in quantities sufficient to run VALID. The design of VALID is compatible with mainframe timesharing, distributed processing, or a mega-micro personal computer. But as we see it, VALID is revolutionary in an educational sense. It is the state of the art in courseware, and it exceeds the grasp of current micro-based systems.

Just as we expect the users of VALID to adapt the course to their own instructional needs, we expect they will be able to adapt to the resource utilization challenges that VALID presents.

References

1. Ager, Tryg A., "APL Tutorials For Elementary Logic," *Proceedings of the Seventh Conference on Computers in the Undergraduate Curriculum*, Binghamton, NY, (1976).

2. Ager, Tryg A. and James McDonald, "Transportability of Computer-Assisted Instruction Programs and Curricula," *Proceedings of the California Educational Computing Consortium Conference*, Anaheim, CA (1978).

3. Garson, James, "Teaching Logic with EMIL," forthcoming.

4. Hearn, Anthony C. and Martin L. Griss, "Software Engineering in the Utah Symbolic Computation Group," Operating Note no. 34, Utah Symbolic Computation Group, University of Utah, 1977.

5. Laymon, Ronald and Thomas Lloyd, *ENIGMA: A Computer Assisted Logic Course*, Columbus, OH: Collegiate Publishing, Inc., 1976.

6. Moor, James and Jack Nelson, "BERTIE: An Interactive Program for Teaching Natural Deduction," *Proceedings of the Seventh Conference on Computers in the Undergraduate Curriculum*, Binghamton, NY (1976).

7. Suppes, Patrick, *Introduction to Logic,* New York: Van Nostrand Reinhold, 1957.

8. Suppes, Patrick, "New Foundations of Objective Probability: Axioms for Propensities," in P. Suppes, L. Henkin, Gr. C. Moisil, & A. Joja (eds.), *Logic, Methodology, and Philosophy of Science IV: Proceedings of the Fourth International Congress for Logic, Methodology, and Philosophy of Science, Bucharest, 1971.*> Amsterdam: North Holland, 1973.

9. Suppes, Patrick, Robert Smith, and Marion Beard, University-Level Computer-Assisted Instruction at Stanford: 1975," *Instructional Science,* 6 (1977), pp. 151-185.

MICROCOMPUTER SOFTWARE DEVELOPMENT:

NEW STRATEGIES FOR A NEW TECHNOLOGY

Kent T. Kehrberg
Minnesota Educational Computing Consortium

The Minnesota Educational Computing Consortium (MECC) has been supporting instructional computing in Minnesota through a large centralized timeshare system. As a result of a task force study in the Spring of 1978, MECC will now also be supporting the use of microcomputers in Minnesota's educational institutions. Through a bid process, the Apple II microcomputer was selected for specific support and software development.

This paper will present the views of the author on the development of software for microcomputers for use in educational environments. The opinions and positions stated are based on experiences with both Apple II microcomputers and the Radio Shack TRS-80 in elementary and secondary educational schools.

CONVERTING TIMESHARING SOFTWARE

Coming from a large-scale timesharing environment, the first temptation is to convert existing timeshare software for use on a microcomputer. To a large extent, this conversion is also what the potential users of microcomputers in Minnesota are expecting. They see the microcomputer as allowing them portability for their computing, releasing their dependence upon phone lines, eliminating their frustrations from busy signals, and providing them with a higher degree of system reliability. They plan to continue using the programs they are familiar with on the timeshare system except now on their new microcomputer.

Because of graphics and tone generation, experienced users of microcomputers (and today experience is measured in days and weeks rather than in years) see the potential for the use of the computer in new subject areas (music, art) and in expanded ways (tutorials). For these people the need for the conversion of existing programs is less. When needed, they can still access the programs through their timeshare system.

If one decides to convert programs from a timeshare system to a microcomputer, the first decision must be whether to do primarily a line-by-line (feature-by-feature) conversion or whether to enhance the program with graphic and sound output. If the program is significantly altered by the addition of graphics, then it may no longer agree with its documentation. Neither will it agree with the previous user's expectations of the program. On the other hand, it may be more interesting and stimulating to the new user.

In general, it is a much better plan to convert an idea from a timeshare system rather than a program. In direct conversion, the addition of graphics and sound may have to be forced--added for visual effect while not completely appropriate. But if graphics and sound are not added to the microcomputer program it will not have appeal. The programming code on the timeshare version may require substantial revision in order to produce a satisfactory version for the microcomputer. Most applications for microcomputers should be developed specifically and uniquely for the microcomputer, using only a pre-existing idea from another computer system.

DESIGNING SOFTWARE FOR USER INTERACTION

Microcomputer software should be designed to take advantage of the unique features of microcomputers. Usually this means making ample use of the graphics capabilities of the micro. Additionally, the developer can consider the use of audio capabilities. Most microcomputers can readily control a cassette tape recorder to deliver pre-recorded instructions or audio drill material. In general, the developer should consider incorporating as many sensual experiences as possible--visual, audio, and tactile.

The microcomputer allows the author to directly involve students in moving through a lesson. The author can have the student point to an object on the screen and then use his finger as a reference point while the computer moves a graphic display around that point. For example, the student could place his finger on the Big Dipper and then the computer could advance the relative position of the stars by one hour. In this way, the student can actively discover the apparent circular motion of the stars about Polaris.

Other tactile experiences can involve having the user record information on a piece of paper. Unless the microcomputer is connected to a printer or disk to record a student's work, there is no evidence to the teacher that the student has run the specified program. Working into the program design some written interaction not only provides a more meaningful experience for the student, but also can serve as a record of the session for the teacher.

IS THE USE OF A MICROCOMPUTER APPROPRIATE?

In the initial design of educational software for a microcomputer, it is important to ask whether or not the learning activity can best be taught with a computer. A popular timesharing program has the computer simulate a pendulum and then allows the student to vary the mass, period, and length in an attempt to discover the effect of each on the period. A much better experience, however, is to give the student a string, paper clip, and some washers. Now with the microcomputer and its graphics capabilities it is again tempting to duplicate this experiment on a machine. Just imagine, the computer can present the pendulum, make it swing, determine the period, and so forth--all in color and with a ticking sound! Still, in most circumstances, strings and washers will provide a more complete and educationally meaningful experience for the student. It is important that we do not attempt to computerize all experiences.

There are special features of microcomputers which do enhance teaching. Computers can simulate events which may otherwise require too much time. They can simulate phenomena which may be too dangerous or too expensive for the classroom. With the graphics capabilities of the microcomputer, an object can be displayed in real-time and then slowed down for careful study. For example, a student can step a four-cycle engine through each piston stroke and watch for the opening and closing of valves and the beginning of the ignition phase.

The precise timing capabilities of microcomputers allows the designer to develop software that can provide students with situations in which responses can be controlled. Math speed drills would not have to vary with the current load on the timeshare system. Motivational typing drills can be developed, and experiments conducted in the area of psychology.

The applications software developer should be concerned about the overuse of graphics in a program. Does every program need entertaining graphics? Will students demand (by their disinterest in a particular application) that software be entertaining? This author's initial concern about this topic was reduced by an elementary principal. The principal stated that students already demand a level of entertainment; that teachers in today's classrooms are competing with Sesame Street. Motivational graphics must be included in microcomputer software programs even for users with no prior computer experience.

DEVELOPING A MICROCOMPUTER APPLICATION

General Considerations

In designing a microcomputer application it is important to know the capabilities and limitations of your equipment. In particular, one must keep in mind that microcomputers do not have the memory size and computational speeds of larger systems. Many times ideas do not have to be rejected but accommodations must be made. Perhaps record keeping will have to be done on paper if files are not available. Students can be instructed to insert programs manually if chaining is not available.

Pay attention to all the possible modes for user interaction. In addition to visual display, it may be possible to add audio output. Students might touch the screen for reference. A supplementary device such as a globe might be referenced as part of a computer-based experience. Application software designers should create programs for active users. Monkeys can push return keys to change the visual stimuli--humans can more fully interact.

Make use of graphics in your program. The quality of the graphics need not be great. Curves made out of little squares still look basically like curves. If a clock has two hands and twelve numbers, students recognize it as a clock--whether or not it is perfectly round.

Plan the role of support materials in the design of application's programs. Can complete instructions be included in the program or should they be available in a booklet or on a sheet of paper? Perhaps some type of worksheet should be used in conjuction with the program. Maybe the student will need to design a strategy prior to coming to the computer. A record sheet for student responses may be appropriate.

Handling Text Material

In the timesharing environment, it was common to ask, "Do you need instructions?", at the beginning of programs. This was generally necessitated by the slow output speeds of teletypes. Few people had the time to wait for the instructions on subsequent runs of the program. On the microcomputer it is not necessary to ask such a question. You can usually display the instructions more quickly than the user can answer the question. If the instructions require more than one screen of output, a few quick carriage returns by the student will move the program on.

Usually some textual material is associated with a computer program. When using text material display only one screen of text at a time. Do not allow the text to roll off the top of the screen. The user should be in control of the display of text. Never time the display anticipating the user's reading rate. Each frame of text should have a standard method of indicating that the material has been read and this technique should be visible on each frame. For example, a simple phrase such as PRESS RETURN may be sufficient.

Authors must pay attention to the readability of the text material. Readability may involve more than difficult words and long sentences. It may also include the size of the displayed characters. Larger characters need to be used for elementary children and those with special learning difficulties.

If the amount of text to be displayed is great, the author should break up the text with questions. Students often tend to skip over the text by entering a few quick carriage returns. It is wise to minimize the amount of text on the screen at any period of time. Do not write text on every possible line.

What is Being Reinforced?

After a few experiences with computers, kids soon learn that certain things happen when they get a question wrong and other things happen when they get it right. Many will deliberately get questions wrong just to find out what happens. Authors must keep in mind that the reinforcement for a positive response should not be less motivational than the program's reaction to a negative response.

One example of rewarding incorrect responses is found in a microcomputer version of hangman (guess the computer's word--letter by letter). This program prints a boat on the screen. If an incorrect response is given, the boat sinks a little lower in the water. If sufficient incorrect letters are chosen, the boat sinks and a little man is seen going down in the water with bubbles rising from his mouth. Very entertaining graphics--considering that you lost! On the other hand, there is very little reward for getting the letters correct and identifying the word. An alternate version might be called torpedo. If you guess the letters in the word, a torpedo is shot at the boat, the boat and the little man sink to the bottom. The more stimulating reward is now for success rather than failure.

The Dynamics Of Display

With the microcomputer, several special techniques should be used when displaying information on the screen. Many of these are needed because of the high rate of speed at which microcomputers output data. Other techniques should be used to compensate for the lack of hard copy on many micros.

1. Screen Formatting. At times it may actually be necessary to slow down the high rate of display on the screen by building a counting loop into a program. This technique will keep the computer busy for a few seconds to allow time for the planned effect. For example, if two similar frames are to be presented, the student may press the RETURN key at the end of the first frame and before glancing up again the second frame has appeared. Thinking that the first frame is still being viewed the student may press the RETURN key again. Clearing the first screen and holding it a few seconds before the second frame is displayed would help emphasize the change.

Any table that may be updated due to student interaction should be continually displayed on a part of the screen. Likewise, if the student has to choose from the same set of options to answer many questions, these options should be frozen on a portion of the screen.

Instructions for continuing from one frame or display to the next should be standardized and highlighted. PRESS 'RETURN' is sufficient but could be strenghtened by blinking or printing in an inverse character mode. Options for stopping a program part way through should also be standardized. A recommended standard term is the word END. Entering this word would bring the program to a conclusion through processing and not to an abrupt halt leaving files open, concluding messages not displayed, and system parameters incorrectly set. STOP is not a satisfactory choice because on many systems this aborts the program similar to typing a break key. QUIT, DONE, and so forth are other possibilities and should perhaps be anticipated in the programming but END is preferrable.

2. User Control Never anticipate a student's reading rate. Unless it is not important that a student read an item completely, don't build it into a timed loop. Rather, allow the student to continue by typing a response. Remember to indicate the response on the screen.

No timed loops should be used for any type of display unless on the same screen directions for continuing are given. Nothing is more frustrating than viewing a display and having no idea about how to continue.

3. Specifying Anticipated Responses The user should not have to guess at the response format expected. If the student is to enter X and Y variables separated by a comma, then print out an example. It would be best to ask for each variable separately.

Be consistent in the nature of responses you expect. Control the movement from one screen of text to another with a standard solicit such as PRESS 'RETURN'. The program could allow any key to be depressed.

It is important not to change the rules on inputting answers. Microcomputers can be programmed to respond to the entry of any single key without the traditional RETURN key. If the computer responds to single keys sometimes and requires a RETURN key at other times, confusion will result on the part of the user. If you want to accept single keys as input then also accept the option of typing the RETURN key. Process the RETURN key as irrelevant input.

4. Screen Overload Beware of overloading the screen. If you segment a display and cause changes in various parts, it can be very difficult for the user to follow. Parts of the display could be static lists of options or some graphic shape. The action part of a display should be centered near the upper part of the screen. If there will be action on another part of the screen, be certain to draw the user's attention to it and allow time for eye movement prior to the event.

Programming Considerations

When it is time to begin programming a unit on the microcomputer, take full advantage of the specific machine being used. If the potential transferability of the program to other microcomputers is a concern, the author will be limiting the use of features on his own system.

The proliferation of microcomputers has lead to many variations in BASIC and any hopes of standardizing BASIC have now been indefinitely postponed. Although the elementary statements in BASIC are common between various micros, the extended statements and graphics techniques definitely are not. The only way to convert graphics from one micro to another is by completely rewriting that portion of the program--generally by looking at the output and not at the code. If you have a general interest in wide use of the software being developed, then choose a microcomputer which has the broadest use in education. At this time the most widely used machines are the 8K PET, 4K Level II Radio Shack, and Apple II using Apple Integer.

An author must remember that the program being developed will probably be run after someone else's program and before yet another program. The last program run may have reset certain system parameters which will interfere with your program: the system's memory may have been specifically allocated, or a graphics screen was left on. These items should be cleared and reset at the beginning of the program. Likewise, authors should insure that a program cleans up at the end--the next program may have been written by someone less thoughtful.

At times a program may have to perform a complicated calculation that may require several seconds or more of compute time. The user should be warned that a calculation requiring time is being performed--wait! Whenever possible the programmer can structure extensive calculation for the time when the user is reading the screen. For example, data could be read into variables while the first page of instructions is displayed.

Special keys on microcomputers will sometimes clear the screen of display material. A student's response, if unanticipated, may cause graphic material on the screen to shift position. Good programming should allow a student to request a redraw of the screen,or the program should automatically check and redraw the screen if necessary.

Test The Program Thoroughly

When a program is complete, the work has just begun. The program should be run with the deliberate intention of bombing it out. Try unexpected inputs. When it works well it should be given to a malicious user. That is, someone who thrives on pointing out the weaknesses in another's program. Finally, the program should also be tested by a naive user--someone who has never used a microcomputer and knows little about the topic of the program. After all three of these test situations the program will still have bugs, but hopefully fewer than before.

Malicious users who are short on imaginative things to do may benefit from a few suggestions: 1) type RETURNS without inputting any data, 2) type in extra long responses, 3) enter numeric data where string data is requested and visa versa, 4) enter two RETURNS in quick succession and 5) enter an option not listed. This series of tests should be tried for each option in the program.

CONCLUSION

Constructing software for microcomputers is a combination of choosing an appropriate topic, applying the dynamics of the system, and programming for every possible kind of user. The suggestions presented in this paper should assist in the appropriate selection of software for microcomputers and in being aware of the care necessary in presenting the material to a student. This combination should lead to new and usable software for microcomputers in education.

MICROCOMPUTER-GENERATED VIDEOTAPES

S.L.Cooke, Jr.; Dept. of Chemistry;
Univ. of Louisville, Louisville, KY 40208
William R. Martin; Martronix Associates;
1235 E. Oak St.; Louisville, KY 40202

INTRODUCTION

Motion pictures and videotapes can provide motion, sound, and color-essential functions in presenting some instructional materials. The reason for the effectiveness of such media is that abstract or formal concepts (1) are best presented using a more concrete (2) medium. Although both have been used, videotapes are superior to film in many ways: more easily produced and modified, reusable medium, more viewing possible before wearing out, easily duplicated, and the equipment is more easily used by inexperienced students. The varieties of video equipment available now or in the foreseeable future have many yet untouched applications.(3)

Video displays are frequently used with terminals and microcomputers. Some of these produce a standard video signal which can be videotaped using conventional video recording equipment. This joining of video and computer technology would seem to lend itself to the direct production of videotapes using a fairly modest system. In the past computer-generated media have generally been done with expensive equipment which usually produced a film. A fairly elaborate system for direct automated media production is now available.(4) A less expensive approach using a video terminal was presented at this conference in 1975 by Adler and Thomas(5) in which videotapes constituted an animated blackboard for foreign language instruction. This work presents an economical and general purpose approach.

The ability of the instructor to produce a high quality, dynamic, visual display for students could be of great benefit to both students and instructors. The techniques involved are no more complex that those suggested for producing animation manually.(6) For the first time, teachers and students can economically and rapidly use the powerful tool of computer animation.(7)

ADVANTAGES

In addition to the general advantages of videotape mentioned earlier, microcomputer-generated videotape has many other advantages:

1) Videotapes can provide the learner with valuable experience to shorten the amount of interactive use of the terminal or computer. For example, the videotape can store a standard or reference computer run for comparison with other runs conducted interactively under learner direction.

2) Videotapes can be used instead of student interaction with the computer in many applications thereby saving computer time.

3) The intelligence of the computer can be used to generate displays and animation as well as numerical results.

4) The videotape can be used in a classroom or an individualized instruction mode and even on home units when it is not possible to have direct learner interaction due to unavailability or malfunction of the computer.

5) It requires much less time to view a videotape than is required of the computer to generate it, thereby saving student time while providing a more coherent learning experience.

6) A sound track, added later, can provide additional information at the right time without requiring the use of the learner's eyes or hands.

7) Computer-generated images may be combined with standard video signals from a video camera using fixed or variable wipe or dissolve techniques.

8) Videotapes not only can be duplicated easily using existing video technology, but also will probably be readily converted to videodiscs when they become available.

DISADVANTAGES:

The production of videotapes by microcomputer also has some disadvantages:

1) The absence of student interaction can reduce student motivation.

2) Program(s) designed for student interaction may not be usable to generate videotapes without modification, thereby requiring additional time and effort of faculty or staff.

3) As technology continues to advance

microcomputers may become more available than videotape players for either classroom or individual use.

MICROCOMPUTER ADVANTAGES:

The slower-than-large-computer speed of the microcomputer is usually not a disadvantage since the speed of communications between computer and video display (commonly 9600 baud) is much faster without the limitations usually imposed by telephone lines or acoustic couplers (generally 300 baud). The marked difference (a factor of 32) opens up many otherwise impossible opportunities, such as fast-moving, high-resolution plots, continuously upgraded tables, and animation.

MICROCOMPUTER HARDWARE

A typical hardware configuration useful for generating videotape recordings is shown in Figure 1. The standard video signal for the operator's console is sent to a videotape recorder. (A preferred alternative is to use a second, independent video display leaving the console free for communication between operator and computer.) The videotape recorder is controlled manually to record the information seen on the monitor.

It is important that these components be mutually compatible, i.e. they should work well together. The direct memory access on some systems precludes simultaneous use of memory by both disk and video interface. It should also be noted that the signals from many video interfaces are not suitable for videotape recording. This is particularly true of color video interfaces. The optional video hardware configuration presented in Figure 2 permits the integration of computer-generated information with scenes viewed by a video camera, as suggested by advantage #7 listed above. The value to the learner is inestimable for situations in which the computer is used to simulate a dynamic process. Examples of such processes include the flow of liquids, the behavior of free-falling objects, and chemical kinetic processes. The highly formal tabulations of the computer can be replaced or reinforced with more concrete and dynamic graphical displays which incorporate moving images showing the behavior of the real objects. The technique has the student using levels much lower on the Knowles (8) cone of experience. Students who are less than formal in their thinking (as defined by Piaget(1)) should find this experience more meaningful and indeed may even become more formal in their thinking as a result.

The capabilities of various video interface circuits are given in Table 1. Table 1 is not exhaustive but the information is as accurate as could be obtained. Note the incompatibilities found in the videotape recorder (VTR) and disk columns.

Figure 3 indicates how such a system might look with the computer responsible not only for computation and display (as in the past and present), but also for controlling the video equipment such as the wipe/dissolve unit or the recorder. Adherence to standards may be the key to transference of current practice into the future.

SOFTWARE

The choice of a programming language for microcomputers for this application depended upon the amount of computation required. Assembly language is usually prohibitive due to insufficient programmer familiarity or time. Interpreter languages offer great convenience for the programmer, but their less efficient operation and the slower speed of the microprocessor combine to yield execution speeds unsuitable for jobs requiring extensive computation.

The preferred language for most applications is a compiler-type language (such as FORTRAN) which, although slower for the programmer, is much faster during execution. Only a few of the microprocessors now available have a compiler language. Table 1 includes a column listing those languages which support graphic displays. These differ in the number and nature of graphic operations available. It is hoped that one of the present industry standards will be adopted and incorporated into the commonly used high-level languages to facilitate both programming and program exchange.

The speed of computation becomes less important if the desired images can be precalculated and stored for rapid retrieval during the actual videotaping. Again, few of the currently available microcomputers have the memory or bulk storage required. A typical black and white (monochrome) image that fills the screen will require from 1K to 8K bytes of memory. An image will typically be displayed for 1/30 to 1 second. The 64K memory limit found on most 8-bit microcomputers yields worst/best time extremes of around 0.2 to 60 seconds. An alternative programming strategy requiring less time and memory (but more programming) would involve storing only the information needed to update each display. A hardware alternative uses bulk storage such as a floppy disk to store a much larger number of images which are merely recalled during the videotaping. A pair of double-sided, dual-density disks would have the corresponding time extremes of around 12 seconds to 6 minutes. It should be noted that color images require more information for the same resolution and hence would yield lower time extremes.

The video editors found in broadcast and instructional television studios can be used to join microcomputer-generated

FIGURE 1. TYPICAL HARDWARE CONFIGURATION

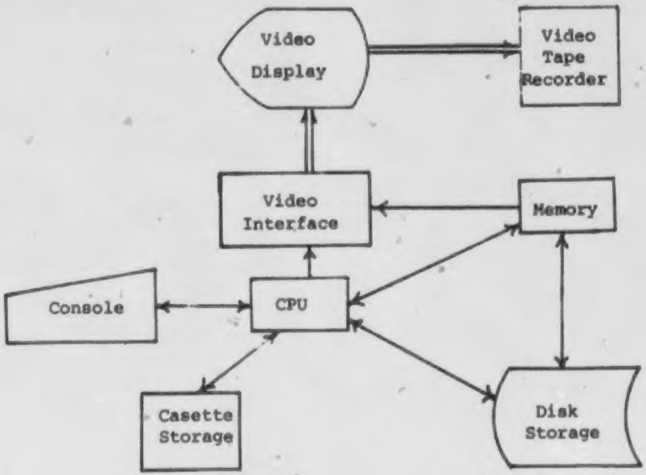
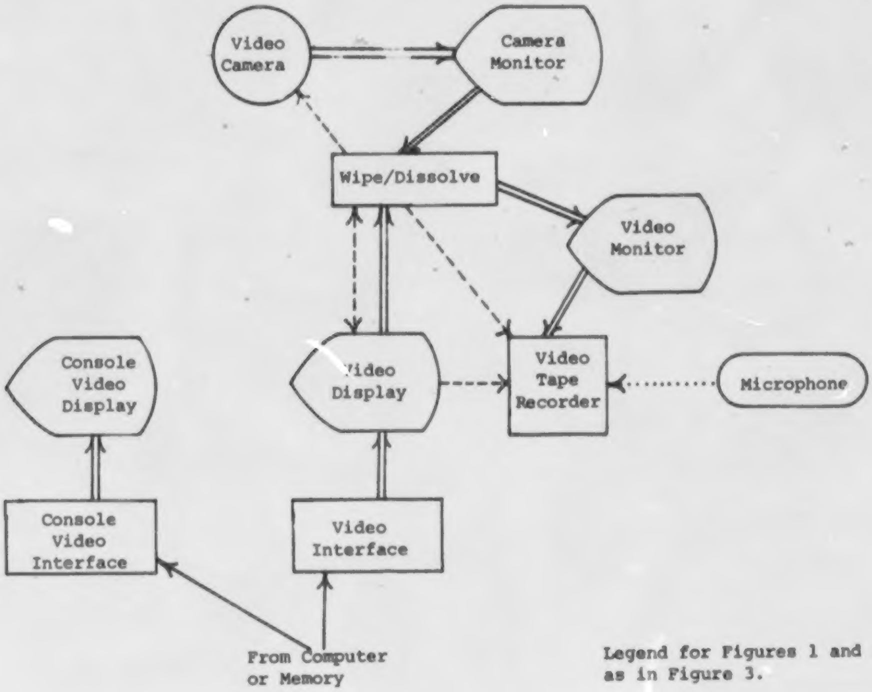

Video Display
Video Tape Recorder
Video Interface
Memory
Console
CPU
Casette Storage
Disk Storage

FIGURE 2. OPTIMAL HARDWARE CONFIGURATION


Video Camera
Camera Monitor
Wipe/Dissolve
Video Monitor
Console Video Display
Video Display
Video Tape Recorder
Microphone
Console Video Interface
Video Interface
From Computer or Memory

Legend for Figures 1 and 2:
as in Figure 3.

FIGURE 3. "FUTURISTIC" VIDEO HARDWARE CONFIGURATION

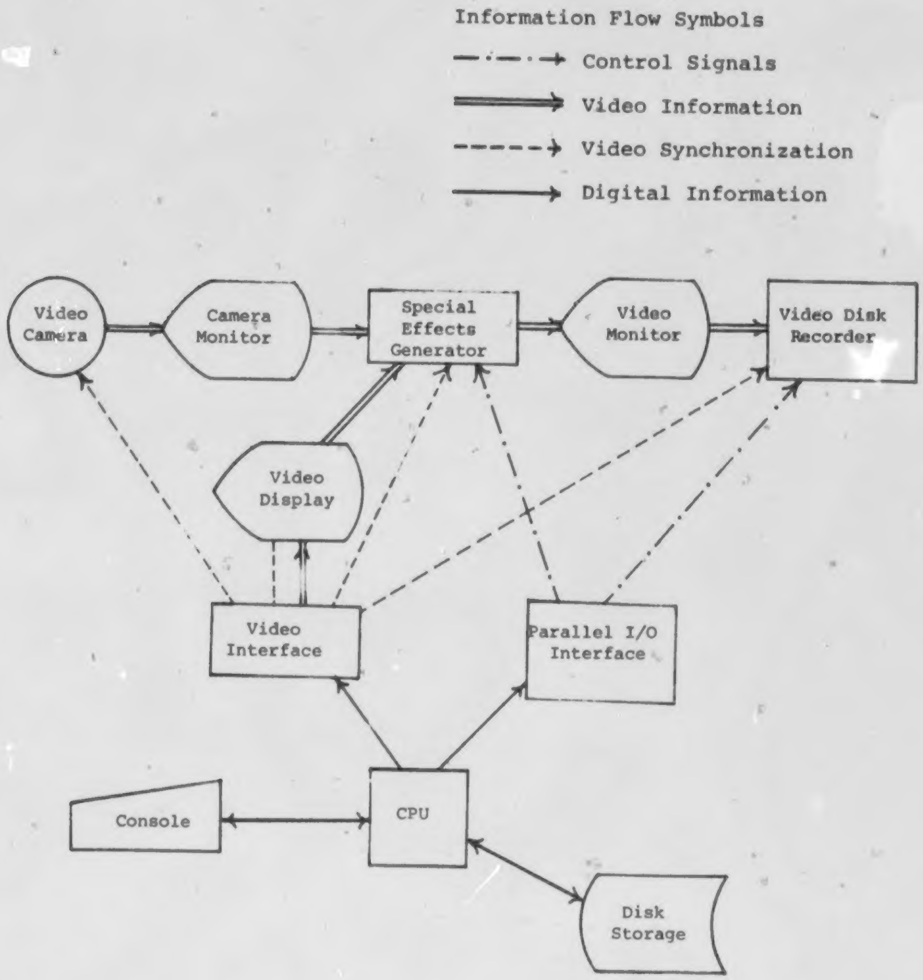

Information Flow Symbols
Control Signals
Video Information
Video Synchronization
Digital Information
Video Camera
Camera Monitor
Special Effects Generator
Video Monitor
Video Disk Recorder
Video Display
Video Interface
Parallel I/O Interface
Console
CPU
Disk Storage

TABLE 1. MICROCOMPUTER VIDEO HARDWARE COMPARISON

Manufacturer	Product	Color or Monochrome	VTR Output (Sync)	Max.Graphic Resolution Hor.x Vert.	Bus	Software Available	Disk
Apple	Console monitor	Color	Mono (no)	40 x 40@ 40 x 48@ 280 x 192@	No	BASIC	Yes
Cromemco	Dazzler	Color	Mono (no)	64 x 64@ 128 x 128@	S-100	BASIC FORTRAN	Yes&
Matrox	Graphic-VRAM	Mono	Yes (in/out)	256 x 256	S-100	-	na
Matrox	Graphic-VRAM	Mono	Yes (in/out)	256 x 256# 256 x 512# 515 x 512# 256 x1024#	LSI-11	-	na
Merlin	Graphics Display	Mono	Yes		S-100	-	na
Pet	Console Monitor	Mono	Yes (out)	50 x 80@ 200 x 40@ 25 x 320@	na	BASIC	Yes
Radio Shack	Console Monitor	Mono	Yes	128 x 48	na	BASIC	Yes
Processor Technology	Console Monitor	Mono	Yes (out)	64 x 16 128 x 48#@	S-100	BASIC FORTRAN PILOT	Yes
Processor Technology	Corona	Color	No	256 x 256	S-100	-	Yes
Sorcerer	Console Monitor	Mono	Yes (no)	240 x 512	S-100	BASIC	Yes
Terak	Console Monitor	Mono	No	320 x 240	LSI-11	PASCAL	Std.
Vector Graphics	High Resolution Board	Mono	Yes (no)	128 x 120@ 256 x 240@	S-100	-	na

* - This information is correct as of Jan. 1979 to the best of the author's abilities based on the information available.
@ - Software selectable; # - hardware determined; & disc and video interface cannot simultaneously access memory; % - some S-100 bus incompatibility; na - not applicable; Std. - standard equipment.

segments together while maintaining the synchronizing pulses which prevent the picture from rolling. The editing process can be very useful for joining a few segments, but it can also be quite tedious if very short segments are to be joined together to make a videotape lasting just a few minutes. The ability of the microcomputer to control the video recorder (Figure 3) would provide yet a third alternative to this problem.

COLOR VS. MONOCHROME

Color and monochrome displays have different advantages. Color provides an added dimension as well as a more pleasing display. The nature of the color video process (three gun tube) necessarily reduces resolution and increases both cost and complexity of the equipment. Thus the choice of color vs. monochrome is dependent upon the application. High resolution plots demand monochrome and do not benefit materially from color. Qualitative and semiquantitative displays would be more effective in color.

Some features of color can be obtained with the higher resolution of monochrome using a grey scale. The gradation between white and black has been separated into distinct shades of grey. Some video interface circuits for graphic displays do not include this feature whereas others typically have either 8 or 16 shades of grey.

WHAT ABOUT THE FUTURE?

The future is certain to bring drastic changes in the areas of computer and video technology. The advent of and the popularity of both the home videotape recorder and the home computer is bound to foster greater compatibility. The prospect of electronic news distribution could ultimately lead to startling developments in electronic instruction. The much lower cost and wider availability of mass marketed appliances can be used to great advantage. The home can become a learning center to supplement, or, when necessary, replace schools and colleges (9,10).

One vital key to the speed, economy, and convenience of this new revolution is the adherence to established standards and the development of new standards as needed. While it is not desirable to establish standards without some experience, it is also true that a less than perfect standard is much better than no standard.

The second vital key to realizing the potential of microcomputer-generated videotapes is software. Modifications must be incorporated into the languages to handle these (up to now) non-standard devices. Subroutines for drawing lines, circles, squares, borders, and axes must be readily available as well as subroutines for animation and synchronized sound generation. The creativity needed to produce interesting and informative materials will become apparent to both teachers and students once the process is made less formidable. The current popularity of super-8 motion pictures has amply proven this point.

The future will certainly be influenced by the continual, rapid changes in computing hardware. Increases in word length, memory capacity, and processor speeds will have an impact on the more demanding applications. More sophisticated color graphics boards should be developed with integrated alphamerics and vector generation features. These hardware improvements may make it unnecessary to resort to a tape medium for many applications.

The fourth and final frontier which is likewise undergoing continual change is video technology. The home videotape unit is here. The videodisc is almost here.(3,10) Computers have been used to generate holograms. Stereoscopic television has been predicted (11). Perhaps someday three-dimensional, full-color motion displays will be generated by computers and stored on an appropriate medium. Certainly the capability to stop or reverse the time axis of a simulation would provide the learner with some of the interaction lost with the direct link to the computer.

SUMMARY

The available hardware and supporting software can be adapted to the direct production of videotapes with a microcomputer. Many advantages can be gained when teachers and students can have this power available at a reasonable price. Specific hardware configurations found useful are included as well as a projected optimal configuration. Recommendations regarding developments for future hardware (both video and computer), software, and utilization are included. A videotape produced using representative techniques will be demonstrated during the presentation.

ACKNOWLEDGEMENTS

The authors would like to acknowledge valuable discussions with Professors Roger H. Geeslin and Thomas M. Jenkins of the Department of Mathematics at the University of Louisville. Partial support from NSF CAUSE grant SER 78-06353 is also gratefully acknowledged.

REFERENCES:

1. Piaget, J. and Inhelder, B., The Growth of Logical Thinking, N.Y., Basic Books, 1958.

2. McKinnon, J.W., "The Influence of a College Inquiry-centered Course in Science on Student Entry into the Formal Operational Stage", doctoral dissertation (U. of Oklahoma,1970) cited in Renner, J.W., et al. Research, Teaching, and Learning with the Piaget Model, Chap. 7, U. of Oklahoma Press, Norman, 1976. (This work describes an effort to expedite intellectual growth in which computers played a part.)

3. Hague, S.M.S., "Video Applications in Education", Educ. Techn., Vol. 18, No. 9, pp. 28-32 (Sept., 1978). Note especially the table on p. 31.

4. Elias, D., "More Individualized Instruction through Automated Media Production", THE Journal,Vol. 5, No. 3, pp. 30-31, May/June, 1978.

5. Adler, A. and Thomas, J-J., "Computer Generated Videotapes for Teaching French Grammar", Sixth Conference on Computers in Undergraduate Curricula, Texas Christian University, Fort Worth, Texas, June, 1975, pp. 13-16.

6. Scheft, T.W.,, "Getting Started in Animation", Media & Methods, Vol. 13, No. 1, pp. 42-44, 61. (Sept., 1976)

7. Halas, J., Computer Animation, Hastings House, N.Y., 1974.

8. Knowles, M.S., The Modern Practice of Adult Education, N.Y., Association Press, 1970.

9. Lewis, R., "Looking Ahead in the U.K.", THE Journal, Vol. 5, No. 5,, p. 59, Sept./Oct., 1978.

10. Bunderson, V., "Futures", Educ. Techn., Vol. 18, No. 4, p. 62, April, 1978.

11. Waters, G.T., "The Impact of the New Techniques on the Television Viewer and Broadcaster", Broadcast Management/ Engineering, Vol 12, No. 3, p. 81, March, 1976.

AN ADVERSE DRUG INTERACTION SYSTEM:
Its Design, Operation and Use

Edwin J. Jaufmann, Jr.
CORPORATE TIME-SHARING SERVICES INCORPORATED

Robert L. Williams
Michael A. Barletta
St. John's University, New York

In the College of Pharmacy at St. John's University an on-line, interactive, adverse drug interaction system has been developed for individual student inquiry, model pharmacy, and classroom instructional uses.

In preliminary planning, several parameters were established. In terms of use, the system would be available to individual undergraduate students generally, and to the model pharmacy, where students practice in a community pharmacy environment specifically. In terms of data, the names of brand and generic drugs needed to be standardized and a coding system developed;drug interaction data, to be researched and collected from primary sources within the College, would prove to be the strength of the system. In terms of operation, it was determined that 1) the system would need to be both economical and efficient in both its mass storage and central processor requirements for execution, and 2) that its searches and displays would have to be executed quickly and efficiently.

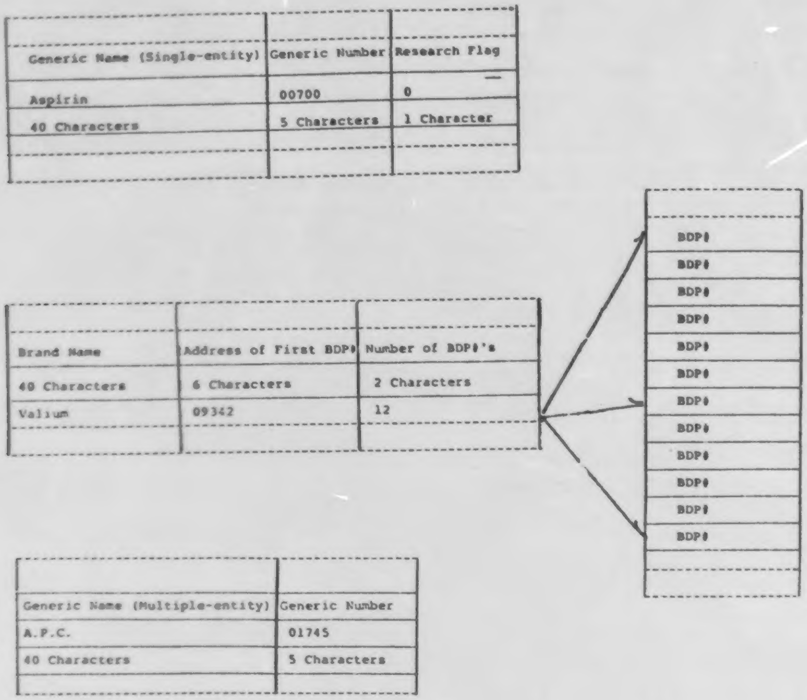
In community pharmacies without computer facilities the pharmacist routinely checks for interaction by manually referencing interaction handbooks. The typical handbook groups drugs into pharmacological families. Thus, the interaction data concerning the interactions of Clofibrate and Coumadin may be found under antihyperlipemic agents and oral anticoagulants, respectively. To assist with searches, indices that cross-reference generic and brand names with family names are usually part of the typical handbook. A thorough search may take a minute or more for the experienced pharmacist.

One problem in grouping drugs into families is that there are many exceptions. In a system using digital (yes-no, on-off) logic there is no convenient way to accommodate these exceptions. Also, within the drug families individual drugs differ in degree of possible adverse interaction with others. Therefore, it was determined that it would be practical not to group by families. It was decided that the system would test single-entity generics against each other. This decision dictated a direction for the consideration of problems concerning drug names, name codes, and file structures.

It seemed readily apparent that to develop our own standards for drug names and codes would prove a monumental task without concern for the more important interaction data to be compiled. The American Society of Hospital Pharmacists' Drug Product Information File (DPIF) proved most advantageous, providing the standardization needed. The DPIF provides standard drug names, both generic and brand, and unique codes for each. It was determined that the standard names and codes of the DPIF would provide a major portion of the data needed for the system, about which our faculty-compiled drug interaction data could be efficiently organized.

A comprehensive set of programs and sort routines was written to convert the DPIF file to a form to be used by the programs of the adverse interaction system. Since the DPIF is equipped with update flags (for periodic additions, deletions, and changes), performing full scale conversions when DPIF revisions are received is not necessary. After conversion, the DPIF becomes a coordinated data base system containing fifteen files. Of these, six files are actually used in the interactive, user-oriented system.

The first set of files in the user oriented system are name files. These consist of the brand name, generic name and multiple-entity generic name files. A diagram of the file structure follows:



Generic Name (Single-entity)	Generic Number	Research Flag
Aspirin	00700	0
40 Characters	5 Characters	1 Character

Brand Name	Address of First BDP#	Number of BDP#'s
40 Characters	6 Characters	2 Characters
Valium	09342	12

BDP#
BDP#
BDP#
BDP#
BDP#
BDP#
BDP#
BDP#
BDP#
BDP#
BDP#
BDP#

Generic Name (Multiple-entity)	Generic Number
A.P.C.	01745
40 Characters	5 Characters

Often there are many brand drug products (with the same name) in several forms with different strengths: they carry different Brand Drug Product Numbers (BDP#'s). For the adverse interaction system, which is only concerned primarily with components of drugs and not with their strengths, one strength form only serves the system's purpose. Therefore, just one BDP# associated with a drug needs to be read and used for additional searches. (Generally, binary or indexed sequential file search techniques for data bases as large as these are suggested.)

Generic drug names — most, but not all, are single-entity drugs — have only one generic code associated with them. This code is the one noted in paragraph four on which the system is focused. As the system is used on-line, when a generic name and its code are found, the primary search is complete when it is a single-entity generic.

Multiple-entity generic drugs also have only one generic number; but, since the drug can and must be broken down into single-entity generic components (e.g. A.P.C. is broken down into aspirin, phenaciten, and caffeine), the system finds its components in a component file. The generic number is not a final step in the search, but a key to an additional search for single-entity generic components that compose multiple-entity generic drugs.

To activate the preceding search, the user inputs two drug names of any type — brand or generic. The first name is then used as a key for the searches. Since brand named drugs are prescribed considerably more frequently than generic — multiple-entity generic drug names are almost never used by pharmacists — the first search is made on brand names (quite often combination drugs), the second on single-entity generics, and the third on multiple-entity generics. A typical user's input follows:

```
INTERACTION? DIMETAPP EXTENTAB,ALCOHOL
     The drug DIMETAPP EXTENTAB has not been fully researched.
```

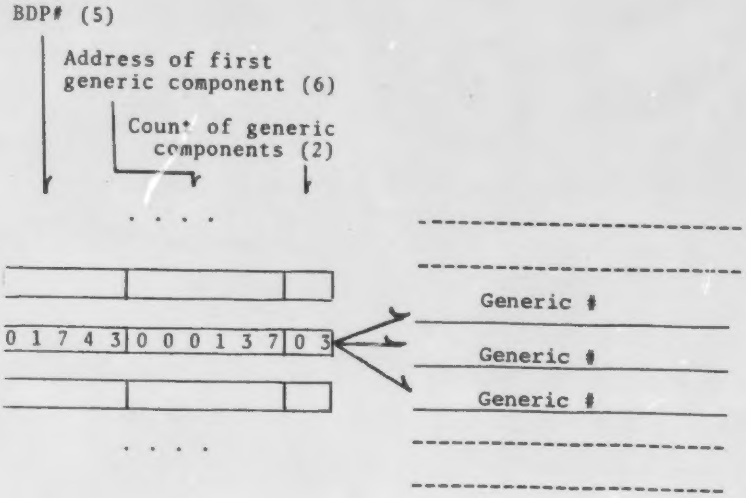
The command line, containing the names, is processed by splitting the string on the comma into two separate names. This method allows spaces to be inserted by the user in the names where appropriate without interference to the system reading of the string.

Upon the successful completion of a name search, further searches and checks are completed, depending on the type of drug the input name is found to be. In the case of single-entity generics, there is no other primary search required. For brand name and multiple-entity generics, a more extensive search routine is necessary.

As previously shown, a BDP# is obtained in the name search. This code is then used as a key for a second search. Although this method appears to be redundant, the information is stored in a manner that efficiently uses the data base system. (Programs that might use BDP#'s directly to describe a drug could access the adverse interaction system data, saving one step in a search.) Although the system's structure requires an additional step, by comparison to its user-oriented model, the savings in mass storage is considerable.

The system searches a second file containing summary information about the drug product including therapeutic category, labeler, dosage form, route of administration, number of generic components, and address of the first component. The number of generic components, the address of the first component, and a determination of whether all of the possible components have been researched are required. The research flag is an important update device in this respect. The system is now conceived to be one that will expand significantly for a period of time. Faculty continually assemble newly researched adverse drug interaction data; as these new data are added to the data bases, the system continues to grow. Through this method, researched information currently in the system may be accessed and presented to the user while indicating that some information has not yet been researched.

A pointer file scheme is employed because the number of component generics contained in multiple-entity drugs normally ranges from two to twelve. An example of this structure follows:


BDP# (5)
Address of first generic component (6)
Count of generic components (2)
0 1 7 4 3
0 0 0 1 3 7
0 3
Generic #
Generic #
Generic #

Because the number of components may range from two to twelve, the flexibility of this scheme, along with its efficiency of storage, is obvious.

Multiple-entity generic drugs are searched in a manner similar to the brand name drugs. The key in this case is generic number, not BDP#. With this exception, there is no difference.

As previously mentioned, the key to this interaction system is a generic-to-generic comparison versus a drug-to-drug comparison. If the search is successful for both user entries, all of the information (the number of and codes for the generic components of both drugs) necessary for the system to conduct the secondary (interaction) search is present.

Although the generic-versus-generic search procedure was followed, a subtle problem involving multiple forms of one generic drug had to be solved. Pharmacologists tend to group ions and salts under one category, as with Warfarin, an anticoagulant. There is no generic drug named Warfarin, but there are nine (9) Warfarin salts and ions which may provide similar possible adverse interactions with another drug. This problem was not difficult for the system itself, but it is one that required consideration by the faculty in their procedures for pharmacological data collection and coding. The solution is that the pharmacologist groups each generic's salts and ions. These — though they may be coded on one form to save the researcher time — become separate records in the system. An example of this follows:

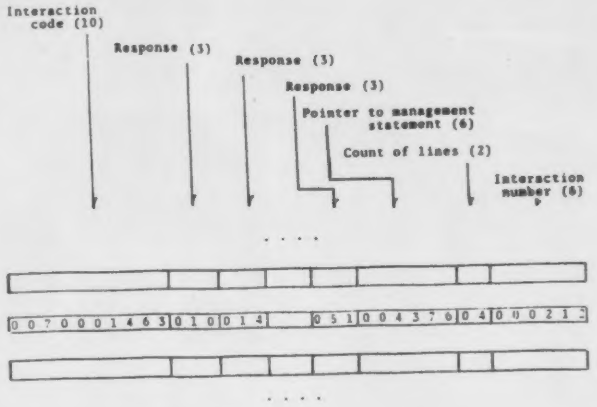
Group name: WARFARIN
Salts and ions: WARFARIN SODIUM
WARFARIN POTASSIUM
WARFARIN SODIUM (CRYSTALLINE)
ACENOCOUMAROL
DICUMAROL
PHENPROCOUMON
ANISINDIONE
DIPHENADIONE
PHENINDIONE

The development of a coding form to be used by the researching pharmacologist was an important step in the development of the system. It proved feasible to divide the effects of an interaction into fourteen categories. Under each category, descriptive responses are checked by the researcher. An unlimited prose management statement about the drug - that is, advice to the user about preventing or lessening the possible adverse effects of an interaction - is also permitted.

In order to both cite the sources of data and validate the research, bibliographic references are also available to the user. References of an example follow:

Year	Publication	Vol.	#	Page
1975	Journal of Pharmacy and Pharmacology	27	11	832
1975	Current Therapeutic Research and Clinical Experimentation	18		560
1974	Journal of Clinical Investigation	53		1607
1973	European Journal of Clinical Pharmacology	6	15	
1971	Annals of The New York Academy of Sciences	179		213
1970	Journal of Pharmaceutical Sciences	59		1258
1970	Clinical Medicine	77	8	20

As shown on the coding sheet, each statement is referenced by a number. These numbers facilitate an efficient use of space on the mass storage device. An interaction record follows:



The code number is a combination of the generic number of a component of a drug present in a patient's regimen and the generic number of a component of the drug which may be added to the patient's regimen. The last three segments of the record are pointers to a clinical management statement and bibliographic references.

The coupling of the two generic codes as illustrated and described above creates a unique, ten-digit, interaction code that is both the key to the interaction search and the code to which the responses by the system are concatenated. The interaction number, arbitrarily assigned to each interaction as completed, acts as an attaching device that relates the interaction records of all salts and ions within the system. This procedure avoids the need for extensive storage for the bibliographic references.

Some (generic) drugs may potentially interact with fifteen or more other drugs. (Where a combination drug is to be assessed against one other combination drug, the number of theoretical assessments to be made by the system can be quite numerous.) Anyone of these, in turn, may potentially interact with many others. A reverse scheme allows the listing, through utility programs, of all drugs which may interact with a fully researched drug. For any two generic drugs, then, coding is produced that gives the desired interaction message. These codes are efficiently manipulated.

An important target during system planning was standardizing codes for the bibliographic sources. After careful consideration it was decided that, since journals were cited in the research, they would be assigned unique numbers. In this one instance, then, we developed our own code system.

As always, procedures to proof data for accuracy are a major concern in the development of an information system. Utility programs can display not only information related to a new interaction but also all previous entries on hard copy for proofing before being incorporated in the system.

Throughout its planning and development potential instructional applications that the system might support were considered. One obvious need was an efficient manual back-up interaction system. Through a utility program an interaction card system is generated and updated as research findings are added to the system. These interaction card files are produced easily, and several are available throughout the College. A second need was a subroutine employing only the drug name search procedure. Because of the hundreds of brand and generic drug names and their various forms, the student should often review drug nomenclature. This subroutine allows the student opportunities to review nomenclature quickly and efficiently. Other subapplications are being considered to support the needs of students and professors.

A full interaction display follows:

```
INTERACTION? COUMADIN,DILANTIN

INTERACTION BETWEEN: WARFARIN SODIUM  & PHENYTOIN SODIUM

 CRITICAL EFFECTS:
      Sedation
      Clotting

 PHARMACOLOGICAL EFFECTS:
      Resulting effects
      of both
      former decreased, latter increased

 CLINICAL SIGNIFICANCE:
      Clinically significant

 MECHANISM OF INTERACTION:
      Multiple

 INTESTINAL ABSORPTION:
      To date ongoing survey does not show

 COMPETITION FOR PLASMA PROTEIN BINDING:
      To date ongoing survey does not show

 METABOLISM OR BIOTRANSFORMATION:
      Both enzyme induction and inhibition
```

```
RENAL EXCRETION:
     To date ongoing survey does not show

MISCELLANEOUS EFFECTS:
     To date ongoing survey does not show

PRECIPITATING DOSAGE OF LATTER DRUG:
     Not predictable

ONSET OF EFFECTS:
     Variable

DURATION OF EFFECTS:
     Variable

CLINICAL RECOMMENDATIONS OF ADMINISTRATION:
     Lower dose of both

CLINICAL MANAGEMENT

Monitor  for   early  signs  of  PHENYTOIN  toxicity  such  as:
nystagmus, blurring vision, lethargy, and poor muscular tone.
Also closely monitor prothrombin times.  Increased anticoagulant
effects develop in approximately 3 weeks.

DO YOU WISH BIBLIOGRAPHIC REFERENCE? YES

                                                    Vol. #  Page
Year Publication
                                                     189 15
1971 Acta Medica Scandinavia
                                                       9 21
1969 Annual Reviews of Pharmacology
```

The system as a whole was not designed to give immediate, summary, life-saving information, although, with minor modification, it could be adapted to a real-life clinical or community situation. The system was designed for use by persons with little or no computing experience as a learning and review tool, geared toward both.

Toward the Development of a Computer-Managed Instructional System in Pharmacy, Based upon Integrated Database System Concepts[1a, 1b]

James G. Henkel* and Linda Wilkinson

College of Pharmacy
University of Minnesota
Minneapolis, Minnesota 55455

School of Pharmacy, Box U-92
University of Connecticut
Storrs, Connecticut 06268

The trend in education over the last quarter century has been characterized by a move toward individualization of learning, not only to provide educational opportunity for everyone, but also to maximize learning by each person.[2] Efforts supporting this trend have taken many forms, from the introduction of behavioral or learning objectives[3] to changes in the form of delivery of instruction, represented by methods such as Programmed Instruction (PI)[4], Individually Prescribed Instruction (IPI), and the Personalized System of Instruction (PSI)[5]. Technology has contributed substantially to these new forms, in terms of both delivery and management of instruction. Applications to delivery include Postlethwait's Audio Tutorial Approach[6] and Computer-Assisted Instruction, while the latter is represented by the Computer-Managed Instructional approaches (CMI)[7].

As outlined by Baker[7], CMI development is usually stimulated by the need to mechanize the greatly increased number of clerical operations in an individualized learning environment, and this need also spawned the project described herein. In the summer of 1974, we undertook the design and implementation of an innovative instructional system in Pharmaceutical Analysis for 130 students in which PSI was to serve as the mode of instruction[8]. Simultaneously, one of us was administering a quarterly course in medical terminology for an average of 200 students, based on a textbook in Programmed Instruction format. Due to the volume of data generated by these overlapping courses and the need for detailed records for research purposes, we sought some form of automated management system. After briefly considering external systems, the decision was made to develop a system internally, based mainly upon the desire to investigate the application of a commercial database management system (System 2000)[9] to the management of instruction. We felt that the insights and decision support capabilities these systems bring to the business and industrial environment could be profitably used in the instructional environment as well, if on a smaller scale. Additionally, we envisioned that the system might serve as the basis for the development of a larger, more complex system that could support a Competency-based Curriculum[10]. Finally, we hoped to consider the instructional process from a different perspective — one based on resource and data management in the corporate sense.

The concept of an integrated database system has evolved from the viewpoint that collectively, the data pertaining to any organization are its most important manageable resource. In fact, the organization itself can be mirrored in, and completely represented by, a centralized store of well-organized data. Moreover, management based upon this store of data is represented to be more effective than that based upon more traditional approaches[11]. In this centralized environment, several important changes may occur, one of which is the integration of the data to accurately reflect the logical characteristics of the organization. A second change is reduction of data redundancy -- each datum is stored in only one place and then shared or accessed in multiple ways. Other changes include concomitant increases in data integrity and security. Using the database concept, it is useful to develop software to organize, control, store, update, and retrieve data in the integrated database. This software, called a Database Management System (DBMS), assumes nearly all of the usual functions of file processing and data storage. With a DBMS, data are addressed by logical addresses rather than specific fields, the logical organization of the data may be modified automatically, and more data may be added without concern for where or how it is physically stored.

The outcome of the above database system approach is that the user gains the ability to retrieve segments of data in reduced or processed form, i.e., as information. Recently, considerable effort has been made to develop and extend the approach to form a Management Information System (MIS). The MIS goes beyond the function of the database system by supporting high level management. Thus, it will perform routine processing functions and more importantly, provide information not easily obtainable by nonautomated means or simple retrieval. By its nature an MIS should be able to support decisions concerning both routine operations and unforeseen problems. Because of this generalizability, a system must evolve with considerable flexibility and cannot be designed fully intact and complete.

---

*Address all correspondence to this author at the University of Connecticut.

Database systems and management information systems have been in use in the educational community for some time, but with few exceptions, their use has been in classical managerial areas, including university fiscal management[12], decision support at the dean or department chair level[13], and management of large vocational training programs[14]. In these contexts, the management functions are applied to student enrollment data, staff and personnel, and control of equipment, facilities and finances. The MIS needs in these situations and the design of better approaches to resource management have been addressed[15-17], as have some reasons why academic MISs can fail[18].

The question of the applicability of integrated database or MIS principles to the instructional environment has not been addressed and we wish to report here our attempts to make such an application. The approach is contained squarely within the domain of computer-managed instruction (CMI) by most definitions. According to Baker[7], a complete CMI system should possess six components: a curricular plan, an instructional model, the capability for diagnosis and prescription management, a reporting feature, and the computer. Elegant examples of computer-managed instructional systems meeting most or all of these criteria have been published[19-23]. When taken in the above database system context, we believe that a complete CMI system should also include an item bank and a facility for objective-referenced test generation, since test administration is the means by which learning is assessed, and test items are certainly a central resource of the instructional process.

By analyzing the instructional environment in the context of an organizational unit with data resources to be centralized and managed, we identified the following discrete but related entities: 1. The curriculum itself, suitably expressed as, and delimited by, competency statements that likely will contain more observable learning objectives. 2. Descriptors of the curricular instructional resources, print and nonprint materials, lecture materials, and drill practice problems. 3. A set of test items associated with each objective in (1); ideally each is validated to measure mastery of the objective. These items represent the domain of the curriculum and its cognitive shape. 4. The student performance data resulting from the administration of the test items in (3). For any student, the profile of item scores may reflect a combination of the degree of knowledge transfer and the uncertainty in the validity of the test items themselves. 5. Student rate of progress data (if meaningful), such as the time taken to complete a learning unit or the number of attempts needed to complete it[5]. 6. Interface data in contact with both the learning and external environments, such as course grades, enrollment lists and transcripts.

Using the principles of database systems design, we have attempted to integrate these data into a database that serves as the central resource for the management of instruction. We expected the resulting management system to serve as a model with which to test the potential of the database approach, as well as to provide a medium through which research and decision information could flow. We hoped that the system would evolve to an effective management information system. Concomitantly we would address other factors, including cost of the system, the hardware resources it would consume, and most importantly, whether or not such a system was even possible using System 2000. To our knowledge, no applications similar to this one had been attempted at the time.

The six data entities mentioned above were divided into two separate classes — resource data and student records. The resource data — the curricular specifications, test items, and learning materials were assigned to one physical database named QUESTIONS, while the student record data were assigned to a second database called RECORDS. An external link was established between the two entities to allow the passage of item analysis data between them. The flow of data and information within the system is shown in Figure 1. Once the learning objectives and test items are loaded into the QUESTIONS database and the registration information is loaded into the RECORDS database, the instructor initiates operation by creating a series of equivalent objective-referenced tests, the profile of which is based upon predefined cognitive and difficulty levels. The student completes a test, which is then graded either mechanically or by hand depending on the environment. The performance scores on the tests are then entered into RECORDS, along with data on the correct and incorrect items. The item analysis data are compiled and electronically transmitted to QUESTIONS. Student performance reports are made available to the instructor who may issue learning prescriptions; if a suitable algorithm is operational, the system may respond to a detected weakness by recommending additional work in one or more areas. Summary statistics are also available to the instructor for both performance and item analyses so that iterative improvements to the system may be made. If resource use is being monitored, profiles of this use may also be generated. By operating in an iterative manner, the usefulness of the system as well as the quality of information it provides should improve as the

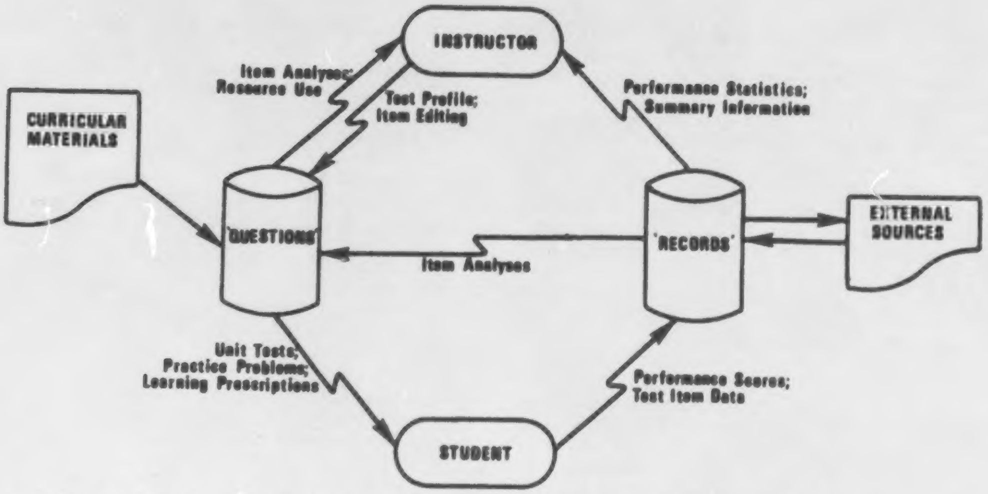

INSTRUCTOR
Item Analyses; Resource Use
Test Profile; Item Editing
Performance Statistics; Summary Information
CURRICULAR MATERIALS
'QUESTIONS'
Item Analyses
'RECORDS'
EXTERNAL SOURCES
Unit Tests; Practice Problems; Learning Prescriptions
Performance Scores; Test Item Data
STUDENT

Figure 1. Data and information flow in the Instructional Management System.

database becomes larger. In most systems, increased size is more of a burden than a benefit. Both databases are maintained and accessed only through the DBMS, System 2000. A database definition, representing the logical relationship among the data elements within the instructional environment, forms the framework within which the system physically stores the data. A representation of the RECORDS database logic is shown in Figure 2. The logic of RECORDS has intentionally been kept very simple and flexible so that it may be used with any instructional model. In the definition, each learning sequence, course, or competency is composed of a variable number of units or objectives. Associated with each of these units is one or more grades or performance scores. To provide greater efficiency of operation and retrieval, the associated grades are maintained not by course but by student. Thus, each course has associated with it a group of students, and associated with each student is a number of performance scores as well as a final or overall score. The date on which each score is achieved is also retained. Although no record is presently kept for learning resources utilized or prescribed, this feature will eventually form a fourth level of data subordinate to each unit grade data element. In the prototype version, the resource feature has been omitted in the order to reduce the system complexity and overhead; also, our operation wasn't equipped to gather the necessary resource use data.

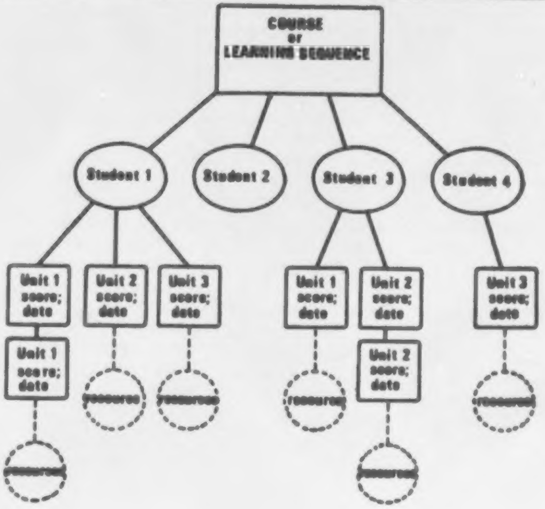

COURSE or LEARNING SEQUENCE
Student 1
Student 2
Student 3
Student 4
Unit 1 score; date
Unit 2 score; date
Unit 3 score; date
Unit 1 score; date
Unit 1 score; date
Unit 2 score; date
Unit 2 score; date
Unit 3 score; date
resources
resources
resources
resources
resources
resources
resources

Figure 2. Logical organization of the RECORDS database (see text).

Conceptually, the QUESTIONS database consists of two parts, the item bank and the curriculum and learning resource descriptors (See Figure 3). The item bank portion is designed to accomodate virtually any form of test item except graphic material. Each test item is associated with a specific learning objective, topical unit, and learning sequence, such as an academic course or competency. As designed, a test item may be associated with only one unit, and each unit is associated with only one course. In other words, a strict inverted tree structure is maintained by the data definition, which is probably not realistic for the higher level objectives or competencies[24]. Although we considered correcting this shortcoming during design and early development, we felt that to incorporate a higher level testing capability into the prototype system would overtax the resources available. Thus, we deferred its implementation to a later time. Presently up to 99 test items may be associated with each objective, and up to 30 objectives are allowed within one topical unit. As many as 20 units per course may be defined. Thus, each course or learning sequence may have as many as 59,400 test items associated with it,

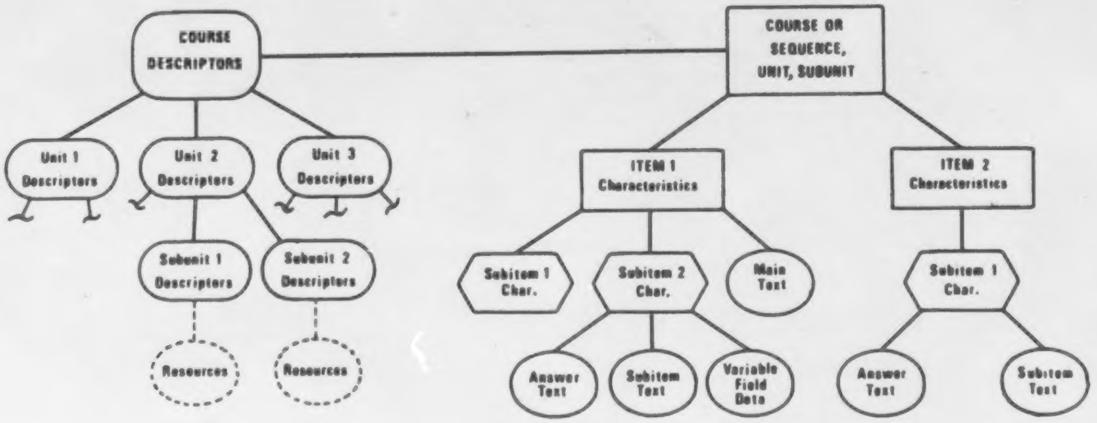


Figure 3. Logical organization of the QUESTIONS database (see text).

representing as many as 600 objectives. We judged these values to be adequate, but if more are needed, the limits can be easily redefined.

Each test item data element has several characteristics associated with it including a unique number, a predefined point value, an indicator of question type (e.g., multiple choice or short answer), and a taxonomic level of difficulty descriptor. No textual data are present at the item characteristics level. Subordinate to this data set are the sub-item and main-text data sets. The sub-item data set contains several other characteristics including control values for item printing, variable field indicators, and analysis data. Subordinate to this data set is the actual item text, the answer text, and if appropriate, a set of variables, one of which is to be inserted at generation time. The main text data set is used only for multipart items. This type of item is characterized by a body of descriptive text presenting a situation followed by several questions or problems pertaining to it. Item 1 in Figure 3 is an example of an item of this type. Most test items are stored in the format of item 2, with the text in one data set and the answer in an associated data set. Because of the heirarchical nature of the data storage, no provision has been made for a feature sometimes called automatic item generation[25]. This feature either randomizes multiple choice distractors or combines isolated phrases to form a randomly generated test item at execution time.The only feature in our system that is similar to this is the use of a variable field to change the numbers in a calculation problem.

The curriculum and resource half of QUESTIONS is defined in a manner identical to the test item classification scheme, with each course or sequence broken down into learning units. Each unit is then further broken down into subunits. Contained in each of these data elements is an alphanumeric course, unit and subunit identifier, as well as a suitably descriptive name of the data set. The alphanumeric identifier corresponds to those in the item bank. The available resources data set, when implemented, will be subordinate to a subunit descriptor, and is shown as dotted lines in Figure 3.

We know of no system yet reported, the design of which is based on modern database system principles. Although functionally similar systems to ours have existed for longer periods of time and are more highly developed at this point[26,27] it has been our hope that the expected advantages of a database pedigree — flexibility, ease of use, and potential for growth into new applications — will be realized, at least in part.

The development of the Instructional Management System has been an evolutionary one, using standard implementation strategies and structured programming where possible. User access to the database system is controlled exclusively by a series of modular COBOL programs, running interactively or in batch mode depending on the function to be performed. Each user function is controlled by one or more separate programs into which is built as much error-checking capability as possible. Thus, separate commands are used to initiate a course or sequence, enter student names and numbers, enter performance data, or modify or delete any data element. Similarly, test items are entered, edited, and deleted, and tests are generated and printed by separate interface programs. In all cases, the user needs to know only the English-like interface commands and is not required to have any knowledge of System 2000 or database techniques.

The real benefit of an integrated database system lies on the output side, especially in its ability to present data in a highly organized, easily readable format, and provide information based upon these data is possible. In the present system, two strategies are used to obtain this material. One is the built-in query/update and report writer feature of the DBMS itself, while the other is a series of modular COBOL programs that interface with it. The query/update feature uses an English-like command language that allows retrieval and formatting of data. All routine reports (e.g., class lists, grade lists, and transcript reports) are made available with no programming effort at all. A particularly useful feature is the system's ability to produce two performance lists, one arranged alphabetically for administrative use and a second containing only student numbers and scores sorted by student number. The second list may be posted, assuring individual anonimity as required by most institutions. The procedure takes about two minutes and can be done as often as needed.

For more complex reports, procedures are used that retrieve ordered summary lists of data upon which one or more COBOL data reduction programs operate. The resulting information is then used to make instructional decisons and monitor student progress. For example, in the PSI-based Analysis course, an important management parameter is each student's progress through the sequence of learning modules. Ideally the instructor should be aware of each person's position in the sequence. With the course mastery requirement set at 90% and the database containing all the information required to produce the daily class progress reports, it was a very simple matter to write a program to print a progress histogram based only on scores at the 90% level or higher. In this way, each student could gauge his or her progress relative to the suggested rate; students who could benefit from help or additional motivation were easily identified by the instructor. An additional listing of those students who had not attempted any quizzes for a number of days was also helpful in this regard. We also used combined summary statistics to schedule proctor resources by correlating days of the week and number of quizzes attempted on each day. While this feature was not developed to the level of a prediction algorithm, it did work nicely in a qualitative sense. For the item bank, after each item had received enough use, a judgement was made regarding its level of difficulty and its consistency with the other items in a given subunit. Again, the summary item analysis data were easily generated using the query feature, and with no programming effort. Many items were modified or eliminated on the basis of these item analysis evaluations. Although the present system requires that the item evaluations be done by hand, our plans are to automate the process to calculate difficulty indices on each item and to report these items that differ from the mean of the items in the objective by a predetermined amount. The significant point of all of the procedures is probably not that they were done, but that they could be done with so little programming effort, using the features available through the DBMS. In this sense, we have confirmed, at least in our minds, that the approach is a viable one that can save substantial human resources by the application of machine resources.

To generate an objective-referenced test, normally the user must specify the course number and a password for the course, unit(s) to be included, and objectives or subunits to be included. One may specify either the number of items to be selected for each objective and the total point value, or only the total number of points on the test. In the latter case, items will be selected randomly with at least one item pertaining to each objective. Provision is made for limitation of the type of item chosen and a taxonomic or difficulty level limitation. All other parameters are self-contained within the database itself. As many different tests as are needed may be generated from any combination of units and subunits, but only within one course. Since the test specifications are generated interactively then automatically submitted to the batch job queue, the generation programs retrieve and format these items, then write the questions and the corresponding answers to two mass storage files for later printing. This procedure both increases security and allows 8½ x 11 inch paper to be loaded on the line printer. Sample tests and answer sheets are contained in the Appendix.

Security considerations in a system such as this are understandably important with respect to both student records and test items and answers. In this regard, the integrated database approach offers a number of advantages over conventional files. Although the system has the usual user account password, if this level of security is broken, the database files still remain unreadable to the unauthorized user. All data are contained in an internalized format on mass storage files and on tape, readable only by the DBMS. If one accesses the files directly, only nonsense characters are generated. Access to the databases through the DBMS is governed by a series of passwords, one for each level of access. Each course data set has unique passwords for both update and retrieval functions, and the appropriate one must be specified at entry. A system access log is kept automatically by System 2000 and it can be consulted to detect unauthorized access if necessary. We have found that the database system approach offers enough built-in security that it isn't necessary to take any of the usual precautions for data protection, such as masking the data.

Another important consideration is the cost of the management system in terms of system resources. The increased cost of systems using this approach has historically been justified by industry in terms of the information they provide to the decision process, thus making the organization more competitive in the market. In many organizations, the increased system costs are also compensated to some degree by increased efficiency and concomitant reduction in personnel. Neither of these justifications is likely to hold in the academic instructional environment. Probably the major arguments on the database system's behalf are the ones of greatly increased flexibility and greater ease of production of management information, both of which we have just begun to experience with the Instructional Management System. Clearly, the purchase of a database management system can't be justified solely on the basis of its instructional management potential, since the $50,000 - $100,000 price is not recoverable in any easily determined manner. Several academic institutions have purchased or are considering purchase of a database management system for general or administrative use, and if the initial cost of the DBMS isn't included in the instructional cost, then the approach discussed here becomes viable and reasonably competitive with other more traditional approaches.

The following discussion of costs of operation is based on figures from system logs of actual use, and does not include the installation and maintenance costs of the DBMS itself or printing costs of $.03 per page. In a system such as this, a direct correlation exists between the depth to which the retrievals and updates reach into the data structure and the system resources used for that operation. For example, adding student names and numbers is more efficient (i.e., less consuming of resources) than updating grades. Similarly, it is more efficient to stay in one branch or area of the database than to retrieve from several branches. In this sense, it is less expensive to generate a 50-point test entirely from one unit than it is to generate a similar test based upon two units. Thus, it is imperative that the design of the database reflect the most efficient access to the data. Retrieval by a less efficient strategy can make a several-fold difference in cost for the identical result. Of course, resource use increases as the size of the database increases. At the Univ. of Minn., charges are made on the basis of a combination of central processing time used and the number of physical records transferred in mass storage. RECORDS processing of student data is inexpensive, with a class of 120 students costing between $3-$4 on the CDC Cyber 74 computer. Performance score data entry, which occurs at a deeper level in the data structure, requires more resources, generally on the order of $8-$12 for the same number of students. Resource usage data would be even more expensive to handle. Retrieval of data is usually considerably less expensive. For example, the PSI class progress chart mentioned above costs about $3 to create, no matter what size the database is.

The test item bank and test generator can be more expensive to operate, depending largely on the tasks it is asked to perform. While a ten-point quiz costs about 33¢ to generate, a larger 50-point test from a much larger item pool costs in the range of $1-$3. Of course, multiple copies lower the cost per student. Typically 8-10 unique tests sufficed for a 200-student medical terminology class at a cost of $15-$20, or $0.075-$0.10 per test per student, which is reasonable. Just as above, update functions consume more resources, and operations such as loading test items, modifying test items, and updating validation statistics tend to be expensive. For example, we have calculated that the approximately 3,500 items in medical terminology represent an investment of about $350-$400 in computer resources, not including personnel time for data entry. This is approximately 10.5 cents per item, and includes modification and validation costs. The smaller item bank for Pharmaceutical Analysis, currently with 564 test items in nine units, works out to about 8.5 cents per item, including correction and validation. This lower cost indicates that a smaller branch requires fewer system resources than a larger one. We view these costs as reasonable for a system this useful and powerful.

There are also overhead costs incurred in storage and backup procedures for the databases which, in our system, cost about $5/week, since the databases are stored on tape and loaded to mass storage only when needed. The cost of interactive terminal connect time also represents resource use, charged at a rate of $0.25/hour of clock time. With connect time averaging 8-10 hours/week for 300 students, the costs add up to $25/quarter. Personnel costs haven't been estimated due to the variable nature of the resource. Often teaching assistants are available or the secretarial staff can provide help entering the data and maintaining the system. A well-trained person in our environment of 300-400 students required about 15 hours per week to perform all the necessary data handling functions. The database system approach is sufficiently complex that its implementation and use should not be attempted without consultation by a competent programmer/analyst working either independently or as part of the computer center staff. Minor but significant adjustments often have to be made as the computer operating system or the DBMS is updated or modified by vendors.

In summary, we present here our experience in the development of a prototype CMI system utilizing integrated database system principles. The Instructional Management System is to be sure still a prototype, with many features yet to be added; based upon early indications the approach appears to be a viable one, worthy of further effort. We have only begun to reap some of the benefits resulting from its use, and while the overhead costs are higher than those of conventional systems, they appear reasonable from a cost/benefit viewpoint. We look forward to the evolution of greater system capability and its extension to other instructional environments. One extension currently being implemented is its extension to a large lecture class in organic chemistry, but results are not yet available.

REFERENCES CITED

1a. Dedicated to the memory of Gordon William Kingston, a true friend whose help and encouragement made this work possible.

1b. Supported by a grant from the University of Minnesota Educational Development Program.

2. For a comprehensive review, see K.P. Cross, "Accent on Learning," Jossey-Bass. San Francisco, 1976.

3. Mager, R. 1962. "Preparing Instructional Objectives." Fearon Publishers. Belmont, California.

4. Silberman, H.F. 1962. Characteristics of Some Recent Studies of Instructional Methods. In: J.E. Coulson, Ed. "Programmed Learning and Computer-Based Instruction." Wiley. New York.

5. Henkel, J.G. 1977. The Personalized System of Instruction, in T.E. Cyrs, Jr., Ed. "Handbook for the Design of Instruction in Pharmacy Education." American Association of Colleges of Pharmacy. Bethesda, Maryland.

6. Postlethwait, S.N., J. Novak, and H.T. Murray, Jr. 1972. "The Audio Tutorial Approach to Learning." Burgess. Minneapolis, Minnesota.

7. Baker, F.B. 1978. "Computer Managed Instruction: Theory and Practice." Educational Technology Publications. Englewood Cliffs, N.J.

8. Abdel-Monem, M.M. and J.G. Henkel. To be published.

9. MRI Systems Corporation, P.O. Box 9968, Austin, Texas 78766. An overview of the operation and features of System 2000 may be obtained in the General Information Manual, published by MRI.

10. Hall, G.E. and H.L. Jones. 1976. "Competency-Based Education - A Process for the Improvement of Education." Prentice-Hall, Inc. Englewood Cliffs, N.J.

11. For a more comprehensive review, see: a) Davis, G.B. 1974. "Management Information System: Conceptual Foundations, Structure, and Development." McGraw-Hill, New York. b) Davis, G.B. and G.C. Everest, Eds. 1976. "Readings in Management Information Systems: A Practical Reference." Q.E.D. Information Sciences. Wellesley, MA d) Ross, R.G. 1978. "Data Base Systems. Design, Implementation, and Management." American Management Associations. New York.

12. Gaither, G.H. 1977. Effects of a Data Base/MIS on University Fiscal Management. AEDS Journal, 10(2):37-48.

13. Powers, M.J. 1978. A Management Information System for Academic Administrators. AEDS Journal, 11(4):97-106.

14. Ohio State University, Center for Vocational Education. 1977. "Management Information System for Vocational Education (MISVE) - General Description." Research and Development Series no. 127. ERIC ED141619 and related material.

15. Schroeder, R.G. 1974. College Management System Improvement: Managerial Topics Part of the Review Phase. ERIC ED119563. See also ED119564.

16. Schroeder, R.G. 1975. "Management Information Systems for Colleges and Universities." Paper presented at the Midatlantic AACSB Dean's Conference, New York. ERIC ED119560.

17. Ross, R.G. 1976. Rethinking Data Base Systems for Education. AEDS Journal, 9(2): 46-51.

18. Hines, G.S. 1977. Why University Management Information Systems have Failed. J. Educ. Data Proc., 14(3):1-7.

19. Kelley, A.C. 1968. An Experiment with TIPS: A Computer-Aided Instructional System for Undergraduate Education. Amer. Econ. Rev., 58(2):446-457.

20. Baker, F.B. Ref. 7, Chapters 4-5.

21. Flanagan, J.C., W.M. Shanner, H.J. Brudner, and R.W. Marker. 1975. An Individualized Instructional System: PLAN*. In H. Talmade, Ed. "Systems of Indivualized Education. McCutchan. Berkeley, California.

22. McManus, J.F. 1971. "Functional Overview of SWRL/IMS Version 3." SWRL Educational Research and Development. Los Alamitos, California.

23. Belt, S.L. and D.W. Spuck. 1975. "Design Specifications for the Generalized Wisconsin System for Instructional Management (WIS-SIM). Working Paper No. 133." University of Wisconsin, Research and Development Center for Cognitive Learning. Madison, Wisconsin. ERIC ED126560, ED126561.

24. Cyrs, T.E., Jr. Personal Communication.

25. Prosser, F. 1974. Item Banking. In G. Lippey, Ed. "Computer-Assisted Test Construction.: Educational Technology Publications. Englewood Cliffs, N.J.

26.Smull, N.W. 1977. "A Profile Examination System for Physician Extenders and a Method for Curricular Prescription. Final Report." University of Missouri-Kansas City, School of Medicine. Kansas City, MO ERIC ED148900.

27. Deutschman, B.W. 1974. Educational Management Information System and Computerized Instruction Support System. In: H.E. Mitzel, Ed. "An Examination of the Short-Range Potential of Computer-Managed Instruction. Conference Proceedings." Chicago, Illinois. ERIC ED112943.

PHARMACY 5210
UNIT 1

TEST 0008
STUDENT NUMBER -----------

NAME --------------------

1. STATE THE MEANING OF THE GREEK WORD PART -PNEA; STATE ITS CORRESPONDING LATIN WORD PART.
2. GIVE THE DERIVATION (G OR L) AND MEANING OF THE WORD PART HELI-
3. GIVE THE DERIVATION (G OR L) AND MEANING OF THE WORD PART UMBILIC-
4. STATE THE MEANING OF THE GREEK WORD PART MEN-; STATE ITS CORRESPONDING LATIN WORD PART.
5. STATE THE MEANING OF THE GREEK WORD PART SEPS-; STATE ITS CORRESPONDING LATIN WORD PART.
6. LABEL ANY WORD ROOTS, PREFIXES, SUFFIXES, AND CONNECTING VOWELS IN THE WORD ARTHRITIS

7. GIVE THE DERIVATION (G OR L) AND MEANING OF THE WORD PART TERMIN-
8. LABEL ANY WORD ROOTS, PREFIXES, SUFFIXES, AND CONNECTING VOWELS IN THE WORD SUPRARENAL

9. GIVE THE DERIVATION (G OR L) AND MEANING OF THE WORD PART CYT-
10. GIVE THE DERIVATION (G OR L) AND MEANING OF THE WORD PART STERE-
11. STATE THE MEANING OF THE GREEK WORD PART CARN-; STATE ITS CORRESPONDING LATIN WORD PART.
12. GIVE THE DERIVATION (G OR L) AND MEANING OF THE WORD PART MAND-
13. GIVE THE DERIVATION (G OR L) AND MEANING OF THE WORD PART -ARTHR-
14. GIVE THE DERIVATION (G OR L) AND MEANING OF THE WORD PART -MALAC-
15. LABEL ANY WORD ROOTS, PREFIXES, SUFFIXES, AND CONNECTING VOWELS IN THE WORD COLPOTOMY

16. GIVE THE DERIVATION (G OR L) AND MEANING OF THE WORD PART AUTO-
17. STATE THE MEANING OF THE LATIN WORD PART PERMEAT-; STATE ITS CORRESPONDING GREEK WORD PART.
18. STATE THE MEANING OF THE LATIN WORD PART CRUS-; STATE ITS CORRESPONDING GREEK WORD PART.
19. STATE THE MEANING OF THE LATIN WORD PART COM-; STATE ITS CORRESPONDING GREEK WORD PART.
20. GIVE THE DERIVATION (G OR L) AND MEANING OF THE WORD PART LIG-

TEST 0004
STUDENT NUMBER ------------

PHARMACY 5410
UNIT 4

NAME ---------------------

1. BRIEFLY DEFINE OR DESCRIBE SYSTEMATIC OR DETERMINATE ERROR

2. UNDER WHAT CONDITIONS ARE CONFIDENCE LIMITS AND T DISTRIBUTION MOST USEFUL?

3. BRIEFLY DEFINE OR DESCRIBE THE T DISTRIBUTION CURVE

4. THREE INDIVIDUALS WERE ASKED TO ANALYZE THE CONTENTS OF A BATCH OF 0.5 PERCENT HYDROCORTISONE OINTMENT. SEVERAL DETERMINATIONS BY EACH LED TO THE FOLLOWING RESULTS (IN MG/G) -

PERSON A - 4.2, 3.9, 4.6, 4.4, AND 3.8 PCT
MEAN = 4.2; S = 0.3; SM = 0.2; RANGE = 0.8; CV = 8.0 PCT
PERSON B - 5.1, 5.3, 5.4, 4.9, AND 5.1 PCT
MEAN = 5.2; S = 0.2; SM = 0.1; RANGE = 0.5; CV = 3.8 PCT
PERSON C - 5.2, 5.0, 4.9, AND 5.1 PCT
MEAN = 5.1; S = 0.2; SM = 0.1; RANGE = 0.4; CV = 3.2 PCT

WHICH DETERMINATION IS THE MOST PRECISE? WHY?

WHICH IS THE MOST ACCURATE? WHY?

GIVE A POSSIBLE REASON FOR PERSON A'S RESULTS.

WHAT WOULD BE THE EFFECT OF INCREASING THE NUMBER OF MEASUREMENTS ON PERSON A'S RESULTS?

5. WHY IS THE COEFFICIENT OF VARIATION USEFUL IN COMPARING DATA OF DIFFERENT MAGNITUDES?

6. AS PART OF A QUALITY CONTROL PROGRAM, A MANUFACTURER TAKES SAMPLES OF ASPIRIN FROM EACH LOT AND PERFORMS STATISTICAL TESTS ON THEM AS WELL AS SPECIFIC TESTS FROM THE COMPENDIA. A SAMPLE OF 20 CHILDRENS ASPIRIN TABLETS WERE ANALYZED BY WEIGHT AS FOLLOWS - 94, 91, 88, 96, 95, 90, 93, 91, 84, 89, 93, 96, 91, 87, 89, 92, 90, 87, 93, AND 94 MG.

CALCULATE THE MEAN TABLET WEIGHT OF THE SAMPLE. SHOW YOUR WORK.

CALCULATE THE STANDARD DEVIATION AND STANDARD ERROR OF THE SAMPLE. SHOW YOUR WORK.

CALCULATE THE COEFFICIENT OF VARIATION AND 95 PERCENT CONFIDENCE LIMITS FOR THE SAMPLE. SHOW YOUR WORK.

EXPRESS THE RESULTS AS THE MEAN PLUS OR MINUS STANDARD ERROR OR CONFIDENCE LIMITS AT THE 95 PERCENT LEVEL. DEFEND YOUR CHOICE OF ERROR LIMITS. COMMENT ON THE RELIABILITY OF THE DATA.

UNIT 4
TEST 0004

1. (04 B11A)
   ERROR RESULTING FROM FAULTY DESIGN; NOT DETECTABLE STATISTICALLY

2. (04 E52A)
   WHEN SAMPLE SIZE IS SMALL (N < 30); RANDOM VARIATION IS WIDER

3. (04 C23A)
   GAUSSIAN CURVE CORRECTED FOR GREATER VARIATION DUE TO SMALL SIZE OF THE SAMPLE; HAS BROADER TAILS AND LOWER PEAK

4. (04 G74A)
   C, BASED ON LOWEST CV AND RANGE

   (04 G74B)
   C, BASED ONLY UPON THE MEAN; C IS ALSO MOST RELIABLE, SINCE IT IS ALSO MOST PRECISE; B A CLOSE SECOND, AND IN FACT WOULD BE HARD TO DISCARD

   (04 G74C)
   POOR TECHNIQUE (SLOPPY); ALSO APPEARS TO HAVE SYSTEMATIC ERROR

   (04 G74D)
   COULD IMPROVE PRECISION AND CORRECT PART OF THE INACCURACY DUE TO BETTER ESTIMATION OF TRUE VALUE; SYSTEMATIC ERROR UNCORRECTABLE

5. (04 D45A)
   NORMALIZES VALUES TO A SCALE OF 100 OR PERCENTAGE, TO GIVE COMMON BASIS FOR COMPARISON

6. (04 F65A)
   SUM(X) = 1823.; N = 20; MEAN = 91 (91.15 UNROUNDED)

   (04 F65B)
   SUM(XΔ2) = 166363.0; [SUM(X)]Δ2 = 3323329.0; S = 3 (3.216 UNROUNDED) CAN ALSO BE 3.2; SM = 0.7 (0.719 UNROUNDED)

   (04 F65C)
   CV = 3.5 PCT (3.53 UNROUNDED); 95 PCT C.L. = (2.090)(0.719) = 1.5

   (04 F65D)
   91 +/- 2 (95 PCT); USE C.L. SINCE N < 30; SM ROUNDS TO 91 +/- 1

DESIGN OF A CAI TOOL TO GENERATIVELY TEACH ETYMOLOGY

Alfredo García[*] and Sara Ann Rude[**]
[*]Computer Science and [**]Psychology
Washington University
St. Louis, Missouri

Introduction and Problem Description

In response to the need of students in biomedical fields for a systematic approach to the formidable language of biomedicine, a sophisticated computer-aided instruction course has been developed. Unlike students of natural languages who learn words that are sysnoymous with words they already know, students in biomedicine must learn words that convey facts, concepts, and relationships that are generally alien to their experience (1). In the majority of cases biomedical terminology comes from Greek and Latin words that are generally combined according to specific rules to describe medical phenomena. A course in medical etymology teaching the techniques of etymologic analysis using several hundred Greek and Latin words (that have common English meanings) may be the key to learning thousands of biomedical terms (2).

Remarkably, the powerful tool for learning the terminology offered by a linguistic approach has been ignored for the most part, even in the light of proven performance (3). Biomedical etymology has rarely influenced existing curricula. Its importance, nevertheless, has not been lost as a rash of do-it-yourself etymology books has been published in the last few years and are recommended reading at many medical and allied health schools (4). An already over-crowded course of studies, combined with irregular schedules and needs that vary from program to program and student to student (2, 4), may have served to motivate curriculum designers to favor self-study (if anything) over a formal course at the beginning of a student's biomedical training.

This problem with its particular set of constraints has led us to the design and development of the Didactic Etymologic Analysis Program I[*] (DEAP I). DEAP is a computer-aided instruction system that draws on computer-monitored instruction, pacing, and scheduling, and on computer-generated presentation of subject matter and exams to teach etymology. The program responds to students' learning needs, time constraints, and intents more efficiently than available programmed texts. Having briefly described the problem, this paper will proceed to specify in some detail 1) the solution requirements, 2) our approach to supporting the solution, 3) the description of the DEAP system, and 4) examples of generative interactions with DEAP.

Solution Requirements

The problem suggests certain requirements of a CAI solution. These include 1) a data store to support interactive teaching of etymologic analysis and 2) provisions for the learning needs of students. These data and learning requirements dictate program design.

The data store provides the matter for the fundamental manipulations in the program. From this data bank the program generates interactions with the student. There are three types of information in this data base: 1) Greek and Latin word elements, 2) the rules and techniques of etymology, and 3) English biomedical terms. The Greek and Latin elements are the building blocks with which etymologic skills are developed. The program must provide a substantial store of Greek and Latin elements with their meanings and grammatical functions as word parts in English terms. The rules and techniques of etymology give the student the means to understand how word elements are combined to form English terms. These must be arranged developmentally in the program so that the student learns to apply them to increasingly complex combinations of Greek and Latin elements. The third input type, English biomedical terms, provide the matter to which emerging etymologic skills are applied. The program must provide a sufficient number of biomedical terms for students to establish a facility for analyzing terms encountered in their field of study. The number need not be exhaustive. It must, however, be illustrative of a wide enough range of Greek and Latin word part combinations to ensure that the student has learned the meaning and use of these elements and can efficiently apply etymologic techniques to unfamiliar words. (For an exhaustive list, the student still has medical dictionaries. The program is not intended to eliminate that need. It is intended to make the use of the dictionary more efficient, meaningful, memorable.) The program manipulates these three types of information in the data store to maximize students' etymologic skill development.

---

[*] The "I" alludes to the fact that it is geared toward Biomedical English as opposed to Common English (II) and Legal English (III).

The learning requirements are designed to respond to individualized needs of students, a major advantage of computer-assisted instruction. For this program four learning needs have been targeted: 1) individual pace through the learning sequence, 2) practice opportunities, 3) specific and comprehensive review capabilities, and 4) evaluation. For maximum learning effectiveness, each of these learning needs should be addressed in two ways: 1) those controlled by the program itself based upon the student's performance and 2) those under direct student concontrol.

The pace of the program presentation should be performance-based. Lesson content size is derived from evaluation of student capabilities. For example, the number of words presented in a single vocabulary exposition depends upon demonstrated performance and is chosen to maximize the material learned in a given time slot. Regular lesson length should be approximately 40 minutes. When performance levels slip below given criteria, the program sequence must be adjustable to meet the student's need for more review or practice of etymologic lessons. These pacing adjustments are controlled by the program itself. Special time-independent lessons may be requested (a vocabulary review, for example).

Practice opportunities need to be built into each lesson in such a way that students use their skills comprehensively during each session and receive feedback. The student should also have the option to request practice opportunities.

Specific and comprehensive reviews need to be provided for the overall regular program sequence. They should also be available on student request. For example, after absence from the program for a period of time, a student may wish to request a review of the last lesson before proceeding with the regular program sequence.

The fourth learning need, evaluation, serves at least two purposes. It provides performance feedback systematically as an aid to both learning itself and motivation. It also provides the statistics upon which construction of future program sequences and pace are based. The evaluation exercises should be identical to those used for practice except that students' performance is logged. On completion of an evaluation segment, students should receive their performance statistics, as an indication of their learning progress. Specific areas of strength and weakness can then be addressed. With these four learning needs met (individualized pace, practice, review, and evaluation) the student should be able to develop etymologic skills flexibly and efficiently.

The solutions that DEAP proposes must provide for each of these data and learning requirements. We feel we have used some creative techniques to meet these requirements and can effectively arm students in biomedical fields with a powerful and efficient tool: competency in etymologic analysis.

Our Approach--Generative Interactions

"Generative interactions" is the term used to denote a question-response-feedback-response-next question type of exchange where the explicit form of this exchange is not preprogrammed. Unlike a programmed-learning text where all the questions and answers are prescribed (i.e., written before the student uses the book or program), generative interactions are exchanges where the questions are generated when the student is at the computer. Only the *type* of questions is determined beforehand. The level of difficulty and the building blocks used to construct the question are determined at generation time. This type of generativity has been studied extensively as applied to examinations (5-11).

DEAP builds generative interactions by selectively tapping the three types of information in the data store. Vocabulary presentations come from the Greek and Latin Elements (GLE). Evaluation and practice are produced using the GLE and the English biomedical terms. Presentation of the concepts of etymologic analysis uses the rules and techniques of etymology along with the biomedical terms and the GLE. The complexity of the generation algorithm is related to the number of data types being used. The algorithms for generating vocabulary presentations are a part of those used for generating evaluations which in turn make up part of the algorithms for generating etymologic techniques presentations.

The generative interactions are the bulwark supporting the solution requirements. They allow students to interact with the program as long as they feel the need to work on a specific technique or to learn a particular set of vocabulary words. They allow the program to offer reviews or representations of materials in new and still challenging ways. This approach also offers students ample opportunity for any additional practice, where their skills and vocabulary may be called upon in a cumulative fashion at varying degrees of difficulty.

Generative interactions, therefore, present a powerful methodology for CAI. Their usage in presenting vocabulary, techniques, and evaluations gives DEAP yet another flexibility not afforded by conventional means. Since the programmed learning in DEAP is not prescribed, one may simply change the terms and the elements, building blocks of the generative interactions, and be able to teach legal English or common English. The rules and techniques of etymology are the same, regardless of the words to which they are applied. Generativity and the design of DEAP thus also provide transformability.

The DEAP System--a Description

When the student logs on DEAP, the program first retrieves his record. The student's record contains cumulative information regarding the student's performance, location in the instruction sequence, a list of problem words, and a list of etymologic techniques suspected of being less than adequately learned. The record also contains specific information regarding dates, length of sessions, progress, and performance at each previous log-on. At this point the student may opt for practice or review/representation instead of continuing the instruction sequence (mode). If students want practice or review/representation, they must indicate the particular lesson (level).

There are five lesson types. Each is a different combination of the basic generative vocabulary, etymologic technique, and evaluation modules. The standard lesson is vocabulary-techniques-evaluation (VTE). The vocabulary presented at any time is illustrative of the techniques being learned. The second lesson type, vocabulary-evaluation (VE), occurs when the quantity of vocabulary may surpass that of the etymologic techniques in a particular location in the instruction sequence. These two types of lessons are geared for approximately a 40-minute session.

The other three lesson types are of variable length and depend on the amount of time the student is willing to spend. In these, the student's performance is not logged. In a practice lesson (PL) the interactions are generated by the evaluation modules. Review/representation of vocabulary (RV) and of the etymologic techniques (RT) are the other two lesson types. In a review, the progress through the material is totally dependent on the student. In representation, a student must satisfy the program's performance criteria before being allowed to continue. (Review is to representation what reading over one's notes after a course is to retaking the course.)

The vocabulary exposition starts by presenting a list of GLE to be learned. The student indicates how many elements are known. That many new elements are added to the list. This augmented vocabulary set hopefully provides a greater challenge than a randomly selected list. The elements from this augmented list are divided into four sublists for presentation to the student. When the student feels a particular sublist is known, an examination is given. Upon successful interaction, the next sublist is presented. After the last sublist has been completed, the student is tested on the entire list. Problem words are presented again to the student and new interactions are generated. In both VTE and VE lessons, the vocabulary list is passed to the evaluation modules for final exposure before the session is over. (See Appendix, Figure I, for examples of vocabulary interactions.)

The evaluation modules compel the student to apply the analysis techniques learned to English biomedical terms while cumulatively calling upon the GLE presented to date. The interactions here are instigated by either a list of GLE to test or by selected biomedical terms. Testing for specific etymologic techniques is a matter of choosing a biomedical term that requires the particular technique for analyses. Techniques the student may have difficulty with are detected in this fashion. On the other hand, testing the student's grasp of the vocabulary is a matter of having him analyze a term containing the element being tested. (See Appendix, Figure II, for examples of evaluation interactions.)

Etymologic techniques are presented by displaying a statement describing each technique. Where a book in etymology would provide examples, DEAP generates interactions. As in vocabulary presentation, the student must interact with DEAP until the program criteria are satisfied. (NOTE: After satisfying the program criteria in the vocabulary and technique presentations, the student may continue interactions until personally satisfied and willing to continue.)

The program itself may request an RV or RE just as the student may. For this DEAP monitors the list of problem elements and etymologic techniques in the student's record. When the elements list becomes too long, an RV is called for. When a particular etymologic technique is misused a number of times, an RE is scheduled. Reviews are also scheduled at the end of major segments of the overall program.

In order to support the generative interactions described here, the three types of information in the data store must have certain characteristics. The GLE must each point to at most 10 terms that contain them. The biomedical terms must each point to all the elements they contain. The biomedical terms must themselves be grouped according to the techniques required for their analysis. (For example, all the words containing only a prefix and a root are located contiguously.) The rules and principles of etymology, along with each descriptive statement, must point to the boundaries of the group of biomedical terms that are illustrative of each particular technique.

Conclusion

Not only does DEAP propose a sophisticated CAI approach that assures learning of the material, it is also geared to take students as far as possible as quickly as they can get there. Generative interactions and the program design offer transformability, as well as allowing DEAP to satisfy a wide range of needs and intents. DEAP is as suitable for the neophyte desirous of a solid foundation as the experienced etymologic analyst intent on consolidating vocabulary or knowledge of explicit techniques.

REFERENCES

1. Bridge, Edward M. The Language of Medicine: A Quantitative Study of Medical Vocabulary. J. Med. Educ., 37:201-210, 1962.

2. Thomas, James B., and Brandt, N. Steele. A Linguistic Approach to Medical Education. J. Med. Educ., 41:697-700, 1966.

3. Richter, W. Study of Anatomy and Ancient Languages: Demonstration of Specialty-Integrated Professional-Language Training Methods (Rostock Model). Verh. Anat. Ges., 71 Pt. 2 1479-84, 1977.

4. Laurence, Jansen, Prof. of Classical Languages and Linguistics, St. Louis University, St. Louis, Mo. (Author of Greek for Scientific Terminology.) Personal Communications, 1976-1979.

5. Ansfield, Paul J. A User Oriented Computer Procedure for Compiling and Generating Examinations. Educational Technology, Vol. 13, No. 3, March 1973, pp. 12-13.

6. Carbonell, J. R. AI in CAI: An Artificial Intelligence Approach to Computer-Assisted Instruction. IEEE Transactions on Man-Machine Systems, Vol. MMS-11, No. 4, December 1970, pp. 181-189.

7. Lippey, Gerald (ed.). Computer Assisted Test Construction. Educational Technology Publications, Englewood Cliffs, New Jersey 07632, September 1974.

8. Olympia, P. L., Jr. Repetitive Domain-Referenced Testing Using Computers. Proceedings of the 1975 Conference on Computers in the Undergraduate Curricula, June 1975, pp. 155-159.

9. Schonberger, Richard J. Modular Instruction with Computer Assembled Repeatable Exams: Second Generation. Educational Technology, Vol. 15, No. 2, February 1975, pp. 36-38.

10. Uhr, Leonard. Teaching Machine Programs That Generate Problems as a Function of Interaction With Students. Proceedings of the 24th ACM National Conference, 1969, pp. 125-134.

11. Whitlock, Lawrence Robert. Interactive Test Construction and Administration in the Generative Exam System. Department of Computer Science, University of Illinois at Urbana-Champaign, August 1976.

APPENDIX

Figure I

Examples of Vocabulary Interactions

```
-Write the element meaning HEART.
#CARD
-Correct
-Which element means SPIRIT,BREATH?
           a-sark
           b-psych
           c-pneum
           d-larngx
#B
-PSYCH means mind, try again
#C
-Correct
-Match the number with the correct letter.
           1-red          a-tom
           2-horse        b-rhin
           3-a running    c-hipp
           a race track
           4-to cut       d-erythr
           5-nose         e-drom
                          f-leuc
                          g-micr
-(answer in succession, corrections are made afterwards)
1-#d
2-#g
3-#e
4-#a
5-#b
-HORSE is not MICR.  MICR means SMALL.  Write the word for HORSE.
#HIPP
-Correct.
```

Figure II

Examples of Evaluation Interactions

```
-Give the meaning for NECROPOLIS
           a-dead throat
           b-throat dwelling
           c-city dwelling
           d-dead city
#C
-The word for city is POLIS, the word for dwelling is ECON.
 Write the meaning of NECR.
#THROAT
-The word for throat is LARNX.
 Write the meaning of NECR.
#DEAD
-Not a bad guess considering there is only one possibility.
 NECR is being placed in the problem words list.
-Write the word that means A MORBID CONDITION OF SLOW SPERM.
#BRADYSPERMATOSIS
-Correct
-Write the word that means a CONDITION OF SHORT FINGERS.
#BRACHYDACTYL
-HMMM, write the word for SHORT
#BRACHY
-Write the word for FINGERS, TOES
#DACTYL
-Write the suffix meaning CONDITION OF.
#IC
-OK, put it all together and write the correct answer.
#BRACHYDACTYLIC
-Correct
-We are going to analyse OMODYNIA
-Write the primary root,
#DYN
-Indicate the meaning of DYN.
           a-foot
           b-hand
           c-large
           d-shoulder
#d
-Correct
-Write the secondary root.
#OM
-Indicate the meaning of OM
           a-small
           b-medium
           c-pain
           d-color
```

#A

-MICR is the word for small. Write the meaning of OM.

#PAIN

-Correct.

-Write prefix.

(CR indicates none)

-Write suffix.

#IA

-Indicate the meaning of IA

  a-study of
  b-condition of
  c-art of
  d-related to

#D

-Good, the medical meaning of OMODYNIA is A PAIN RELATED TO A PROBLEM IN THE SHOULDER.

Figure III

Examples of the Etymologic
Techniques Presentation

-Prefixes ending in N change the N to an M if the root following the prefix begins with a p, b, or ph.

-eg.1 EMPATHY

-Write the root.

#PATH

-Write the prefix.

#EM

-EM is not the real prefix.

#EN

-Correct. Write the suffix

#Y

-Correct. EMPATHY means a FEELING IN.

-eg.2 SYMBIOSIS

-Write the root.

#BI

-Write the prefix.

#SYN

-Write the suffix.

#OSIS

-Correct. SYMBIOSIS IS A CONDITION OF LIVING TOGETHER.

COMPUTER TRAINING FOR THE FACULTY AT DREW UNIVERSITY

Dr. J. A. Metzler, Mathematics Department, Drew University, Madison, NJ 07940
201/377-3000, ext.327

Dr. G. V. Nelson, Chemistry Department, Drew University, Madison, NJ 07940
201/377-3000, ext.368

Dr. W. J. Carroll, Economics Department, Drew University, Madison, NJ 07940
201/377-3000, ext.327

1. INTRODUCTION

Drew University in Madison, New Jersey, comprises an undergraduate college of liberal arts, a graduate school, and a seminary. The total enrollment of the University is approximately 2,200 students. Of this number, 1,650 students are in the college of liberal arts, 250 are in the graduate school, and 300 in the seminary.

Until the summer of 1978, Drew's computational needs were met by operating in-house an IBM 1130, and by accessing the New Jersey Educational Network (NJECN). The IBM 1130 was operated primarily in batch mode. The NJECN was accessed by two terminals and by a remote job entry station. For the academic year 1977-1978, the University allocated $15,000 for NJRCN usage.

As a response to a growing institutional awareness of the need to more fully integrate computing into the curriculum, the Academic Computing Committee (ACC) was formed. One of the first accomplishments of the ACC was to establish goals for computing. These goals were that:

a. Drew graduates have a basic understanding of what computers are and are not, and what they can and cannot do;

b. Drew graduates have an understanding of the general role and use of computers in their major area of study;

c. Some Drew students would acquire the necessary knowledge of computers to understand, develop, and evaluate computer applications in their major area of study.

In order to intelligently administer the proposed integration of computing into the curriculum, Dr. Nelson conducted a survey of the present and proposed academic uses of the computer within the University. The survey uncovered a number of important considerations outlined in the next section. One of the considerations strongly expressed was the faculty's desire for some training in the use of computers. It is the purpose of this paper to describe a course designed and implemented by the authors to provide the faculty with this training.

2. THE SURVEY

The survey was conducted in the spring of 1976. Dr. Nelson interviewed designated departmental representatives from each of the departments of the College of Liberal Arts as well as some of the departments in the graduate school and seminary. The results of the survey summarized the current academic uses of the computer and revealed faculty attitude toward any anticipated expansion in computing.

The current academic uses of the computer were organized in terms of computer training, instructional uses in support of other courses, and research uses. Computer training included a regularly scheduled freshman-level course entitled Introduction to Computer Science and a small number of occasional courses. These courses usually used the IBM 1130.

Instructional uses in support of other courses made use of both the IBM 1130 and the NJECN to:

a. perform numerical computations (mathematics and natural sciences);

b. statistically analyze compiled data (social sciences, natural sciences, and the humanities);

c. simulate and graphically present models or examples of complex situations (chemistry, economics, psychology, zoology);

d. retrieve information from data files.

Research uses of the computer included both student as well as faculty research. Student research typically involved detailed analysis according to research methods learned in a departmental course. Faculty research was segmented into either course-related research, funded research, or personal research. Research projects were done on either the IBM 1130 or through the NJECN depending upon the application and the availability of funding.

The attitude of the faculty towards the expansion was somewhat surprising. The budget for Drew University, like most colleges and universities, had been going through a period of minimum growth. Hence, any expansion of services in one area threatened a budget reduction in another. Also, given the nature of the University, a great majority of the faculty were in fields removed from any form of technology. For these reasons, Dr. Nelson prepared himself to receive considerable adverse reaction.

In general, however, there were no adverse reactions among faculty members to the notion of expansion of computer services for either administrative or academic use. In every case, the faculty were willing to talk, and if not computer users themselves, they were aware of computerized developments in their areas. The general indication was a definite support of computing.

The faculty desired proposed uses of the computer to follow past guidelines. What the faculty did request were significant departures in terms of training faculty, convenience in operation, and professional staffing support leading to more efficient use of computing both in courses as well as in student and faculty research.

3. THE PRESENT SITUATION

In the summer of 1978, Drew University installed an HP-3000 computer system with sixteen terminals. In the spring of 1977, the University received a $150,000 grant from the Mellon Foundation for faculty development. In response to the survey results, approximately $5,000 of this money was allocated for the training of the faculty on the new system.

As preparation for this training, the authors attended a five-day course, Introduction to the HP-3000, offered by Hewlett-Packard, and spent an additional four months becoming familiar with the HP-3000's applications in coursework. Finally, some time reviewing and organizing led to the series of mini-courses outlined in the appendix to this paper. The specific goal of these mini-courses was to impart to the faculty:

a. an understanding of what computers are and what they can do;

b. an exchange of ideas and information on how computers can be used in the classroom;

c. an understanding of how the HP-3000 differs from the IBM-1130;

d. an introduction to programming and programming languages;

e. a familiarity with HP-3000 Editor commands;

f. the ability to adapt a pre-existant program to the HP-3000;

g. the ability to use SPSS on the HP-3000;

h. enough confidence in each of the participants so that he or she can go on to learn more.

The course was offered in January and repeated in February 1979. It was taught by the authors and Dr. William Carroll who did the session on SPSS. Forty faculty members, including two deans, from throughout the University enrolled in the course. An evaluation of the course is presently being done. The results of this evaluation will be discussed at the conference.

APPENDIX

Introduction to Computing at Drew January 8 February 12 This course will present a general overview of Drew's new computing facilities and modes of faculty training. The topics covered include: uses of computers in course work, tour of the HP-3000 facilities, demonstrations of HP-3000 capabilities, introduction to other courses to be offered and other possible modes of training.

Using the Editor on the HP-3000 January 9 February 14 The Editor is the basic tool for interactively communicating with the HP-3000. The course will develop most of the major capabilities of the Editor. The topics covered include: signing on to the computer, the difference between batch processing and time sharing, basic and advanced Editor commands, an introduction to MPE system commands.

Introduction to Programming January 11 February 19 It is anticipated that most Drew faculty members will not be spending a lot of time programming; however, a solid knowledge of what a program is will greatly increase a faculty member's ability to use the computer. BASIC (Beginners All Purpose Symbolic Instruction Code) is a simple, versatile language in which many educational applications are written. The course will develop the elements of programming using the BASIC language on the HP-3000. The topics covered include: storing data, iteration, simple logic, input-output, and adapting other programs to the Drew computer.

Using the HP-3000 for Input/Output January 15 February 21 The course is designed to demonstrate how to run a pre-existant program on the HP-3000. The topics covered include: creating a program file, creating a data file, creating a stream job, important MPE commands, sorting a pre-existant deck into a file, and adapting other programs to the Drew computer.

Introduction to SPSS (Statistical Package for the Social Sciences) January 18 February 26 SPSS is a package of computer programs generally designed for the modification and analysis of social data. It may, of course, also be used for the analysis of other types of data.

The course will explain how to gain access to SPSS on the HP-3000. Topics covered include: description and modification of data sets available; derivation of common statistical measures such as the mean, standard deviation, and the variance; regression and correlation analysis. No prior knowledge of statistics or computing is required for this course.

Transitioning Instructor Roles In Computer-Based Environments: A Preliminary Program for Improving Instructor Attitudes

Jacqueline Lea Dobrovolny,
Barbara L. McCombs & Wilson
A. Judd
McDonnell Douglas Astronautics
Company - St. Louis

The use of self-paced individualized instruction, and particularly computer-managed instruction (CMI), is becoming an important part of cost-effective military technical training. There are, however, many aspects of such systems which are creating new and challenging problems. As the developers and users of large-scale CMI systems become familiar with unforeseen problems in the implementation of such systems, it becomes clear that there are many factors to be considered and much to learn before the full potential of CMI can be realized.

For the purpose of this discussion, the total CMI system can be divided into two components: (1) a hardware/software/courseware component, and (2) a human component (the classroom instructors and instructional management hierarchy). To date, CMI research and development efforts have concentrated almost exclusively on the hardware/software/courseware component. These efforts have resulted in substantial advances in the CMI technology, and have indicated a strong potential for CMI in technical training environments.

Many of these advances, however, have been at the expense of the human component, particularly the classroom instructors. That is, CMI instructors have been forced to relinquish their role as dispenser of information, evaluator, and essentially the center of the educational process--a role which has considerable prestige and power--and to substitute in its place the role of counselor, diagnostician, tutor, and remediator. This latter role can be as emotionally rewarding to properly trained instructors as their earlier, conventional instructor role. If, however, instructors do not have the appropriate skills to perform their new responsibilities, the CMI role will provide them with minimal, if any, ego fulfillment. As a result, many untrained CMI instructors believe that they are little more than clerks in the learning centers, and this feeling of impotence is often manifested in hostility toward every aspect of the CMI system. This problem is often compounded by the failure of management personnel to recognize the instructors' feelings of frustration and to provide CMI instructors with a replacement role which is as satisfying and rewarding as the one which they have relinquished.

An example of this problem was encountered during the development and implementation of the Air Force Advanced Instructional System (AIS). The AIS is a prototype, multi-media, computer-based instructional system designed to improve the effectiveness and efficiency of Air Force technical training and to provide an operational research facility for assessing innovations in instructional technology. The system supports four technical training courses representative of the range of cognitive and performance skills required by enlisted Air Force personnel. An adaptive instructional decision model uses state-of-the-art computer hardware and software, as well as currently available statistical methodologies and instructional procedures, to provide instructional management and individualized assignments to alternative instructional materials.

Problems resulting from the lack of specific CMI instructor role definition and training became particularly obvious during subsequent instructional efforts within the AIS system. Near the end of the AIS contract period, a CMI Study Skills Project (McCombs, Dobrovolny, Judd, in press) was begun to (1) determine the characteristic problems which students encounter in a CMI environment and those strategies which help students cope with these problems; (2) design, develop, implement, and evaluate a small set of self-contained instructional modules for increasing the effectiveness with which students adapt to and perform in a CMI environment, and (3) investigate procedures for individualizing assignment of these modules so as to minimize training time and cost. In the process of investigating the first two goals, the evaluators discovered that most instructors did not know how to perform the role of an effective CMI instructor and were, therefore, quite ambivalent about their new responsibilities. Additionally, it was apparent that while most students needed assistance in improving their study skills, many instructors knew little more about effective study techniques than did the students.

Given this situation, it was agreed that the instructor role problem was of sufficient importance to merit investigation in place of the individualization goal. Thus, an Instructor Orientation and Training Package was designed, developed, and evaluated. These activities are the focus of this paper.

Related Research

In analyzing the AIS instructor role problem, it became apparent that the problem of training instructors for their new CMI roles is not unique to AIS. For example, the importance of adequately training instructors in their new roles in computer-based environments and the existing need for such training programs has been pointed out by a number of researchers (e.g., Hansen & Harvey, 1970; Hursh, 1976; Stanchfield, 1974; Wilson, 1973). The seriousness of this situation is further reflected in Hursh's (1976) discussion of effectiveness data for personalized systems of instruction (PSI), and his conclusion that the effectiveness of CMI-related instructional systems such as PSI is related to the adequacy of instructor training in the specific behaviors required.

Specific delineation of the range of instructional roles required in a computer-based environment are outlined by a number of researchers, (e.g., Bunderson, 1970; Campbell, 1977; Dyer, 1972; Hansen & Harvey, 1970; Harvey, 1978; Kooi & Geddes, 1970; Lamos, 1971; Stanchfield, 1974, Wilson, 1973). There is fairly general consensus that new roles required of instructors in computer-based instructional environments include: (a) diagnostician of individual student learning problems; (b) prescriber of special learning activities; (c) manager of instructional resources, events, of student progress; (d) learning psychologist, counselor, and advisor, (e) subject matter expert; and (f) author of instructional software. Additionally, Falzetta (1973) points out that ideally, the computer frees the instructor to assist students in becoming more skilled and responsible human beings. The instructor's role becomes one of guiding the student to a love of learning and, as such, provides needed humanization qualities in this new environment.

Wittrock and Lumsdaine (1977) indicate that in an environment in which learners are responsible for their own learning, the instructor's role must become one of being responsible for changing the learners' inappropriate attributions of success or failure, and helping them attend selectively to the information to be learned and to construct meaning from it. That is, a basic role of the instructor in a CMI environment is one of facilitating student acquisition of the active information processing strategies and skills that will allow learners to take responsibility for their own learning.

One example of a training program of relevance to instructor skills required in a CMI environment is provided by Cohen, Emrich, and deCharms (1977). Teachers were taught to differentiate among four possible teaching styles and were encouraged to use those styles which fostered more positive personal causation skills in students. Elements of such teaching styles included a flexible approach to learning that stimulates problem solving behaviors in students, exploration of learning alternatives, and active, cooperative classroom activities. Results of this study indicated that whereas teachers demonstrated knowledge of the styles, they did not later transfer the behaviors learned in the workshops to the classroom. Cohen et al. argue that the reason for their findings was that one cannot assume that because someone knows new skills, they also want to, and know how to, apply these skills. Thus, it is stressed that any successful teacher training program cover three components--that it move beyond merely cognitive understanding to the want to, and try to dimensions.

In the deCharms' (1976) study, teachers taught their students that they were origins rather than pawns and that they could influence their learning outcomes and destiny. The program emphasized student self-concept, achievement motivation, realistic goal-setting, and the concept of origin-pawn. Results reported for a two-year period were that the program increased the motivation of both teachers and students and was directly related to improvements in academic achievement.

Two programs which support the utility of workshop formats for training teachers in new skill requirements were reported by Cadenhead (1976) and by Rash and Grimm (1976). Cadenhead describes a training program in which teachers were successfully taught four primary roles in small-group workshop sessions: diagnostician, facilitator, interactor, and innovator. Rash and Grimm, in describing implementation of a self-directed learning program for junior high school students, discuss a preservice training program to prepare teachers for the changes they would have to make in instructional practices. This program included team planning activities in which teachers practiced acting as student learning advisors and planning learning activities, establishing student performance criteria, and planning strategies to increase the effectiveness of each student's learning program. The positive results obtained by these programs thus support the importance of using (a) group process variables to stimulate instructor involvement and interest and (b) practice exercises to insure the transfer of new skills to the instructional environment.

Four tentative conclusions drawn from the literature were:

(1) Definition of CMI instructor roles should focus on those roles essential to the quality of the student-instructor interaction, and should consider the ideal instructional capabilities of computer-based systems in deriving meaningful and theoretically sound instructor roles;

(2) Cognitive models of the learning process, and particularly Wittrock's work in this area, appear particularly promising as a theoretical framework for defining the instructor role of facilitator of student learning;

(3) Instructor training in those skills intended to facilitate student learning should focus on those roles most directly involved in the student-instructor tutorial relationship, e.g., those skills required in their roles as diagnostician, tutor, and prescriber of instructional events;

(4) Instructor training should include those procedures and strategies most conducive to the formation of positive instructor attitudes and the transfer of their new skills to the training environment, e.g., a workshop format to foster positive group processes, active participant involvement, and representative practice of skills.

Method

At the beginning of the CMI Skills Modules Project, student interviews were conducted to determine the characteristic problems which students experience in a CMI environment. The major result of these interviews was the design, development, and evaluation of a CMI "Orientation/Time Management Lesson" (McCombs, et al., in press). This lesson discussed learner responsibility in a CMI environment as recommended by Wittrock and Lumsdaine (1977) and specific behaviors required by this responsibility. The lesson also provided students with rudimentary time management skills.

An equally important result of these interviews was the observation that instructors did not know how to perform the role of an effective CMI instructor which appeared to lead to their ambivalent feelings about their new responsibilities. Additionally, although most students needed assistance in improving their study skills, many instructors knew little more about effective study techniques than did the students. Thus, a second set of student and instructor interviews were conducted in order to more clearly define those skills in which instructors needed training and those study skill problems which were most difficult for students.

The results of these interviews formed the basis for the design, development, and preliminary evaluation of an Instructor Orientation and Training Package. This package contained a three-session set of instructor workshops, separate skill training in the areas of Reading Comprehension, Memorization, Test-taking and Concentration Management (Study Skills Modules), and a diagnostic/evaluation tool entitled the Study Skills Questionnaire (SSQ). The workshop package was designed to improve the functioning of not only the human component of the total CMI system, but also the interaction between the human component and the hardware/software/courseware component.

In accordance with the Cadenhead (1976) and the Rash and Grimm (1976) studies, the instructor workshops were designed to use group process variables to stimulate instructor involvement and interest and to provide a setting for practice exercises. It was also deemed important to provide instructors with the opportunity to discuss and share their uncertainty about their new role in a computer-based instructional environment. Thus, each workshop session was carefully planned to enable instructors to have numerous opportunities to discuss their concerns.

The first session of the workshop focused entirely on the practice and discussion of the techniques and strategies provided in the Study Skills Modules. This session lasted approximately one hour and was concluded with a homework assignment of having all participants make a list of those clues that they used to identify students in need of assistance.

The second session of the workshop also lasted approximately one hour and was devoted to the practice and discussion of problem solving skills, diagnostic strategies, and remediation procedures, including some training in listening and probing skills. The SSQ, which was designed to assist instructors in identifying students in need of study skills remediation and to evaluate the effectiveness of this remediation, was also discussed and explained. Basically, the questionnaire assessed each student's opinion of his/her study skills in each of the four study skill areas. Instructors could access individual student Study Skills scores via an interpretive display available on an AIS interactive terminal.

These two workshop sessions were held on consecutive days. Following the second session, instructors were given the responsibility for identifying students in need of assistance and prescribing appropriate remediation. This responsibility included the assignment of the Study Skills Modules to students who needed remediation in one or more of the four areas. Each of these four modules was packaged individually so that an instructor could assign any or all of the modules separately, and these materials could then be retained by the students throughout the course. Although the Study Skills remediation assignments could have been determined statistically by other personnel or even by the students themselves, it was decided, based on much of the literature, that giving instructors this diagnostic responsibility was vital to the development of positive instructor roles.

The third session of the workshop also lasted approximately one hour and was designed to capitalize on the effects of group process variables and, as such, to give instructors a chance to discuss with each other and the group leaders any problems they had experienced in using their new skills and to exchange solutions, suggestions, and interesting case histories. Instructors were also asked to complete a twenty-item critique during this session which was designed to measure instructor attitudes toward the workshop procedures.

In order to insure that instructors participating in this program were committed to the importance of instructor training, only volunteers were used for the two evaluations. These evaluations consisted of both a small group tryout and an operational tryout. In conjunction with the small group tryout of the instructor workshop, the SSQ was entered into the hierarchy of each course as a requirement for all students after they finished the first block of their course and again before they began the last block of their course. Thus, after the instructors completed their training, there were already some students in their learning centers with potentially useful diagnostic information.

The small group tryout of the instructor workshop was conducted with four instructors from the Materiel Facilities (MF) course and four instructors from the Weapons Mechanic (WM) course. The purpose of the small group tryout was to insure that the content and format of the workshops were appropriate prior to operational tryout. The results of their critiques indicated that they were generally very satisfied with the workshop procedures. Those activities which were most popular were the explanations of the techniques presented in the four Study Skills Modules and the diagnostic question handouts and explanations. Other activities rated highly were the exercise in listing internal and external clues to student study skill problems, the examples of when communication problems occur, the passage on Attending Behavior, and the discussion of reflection and probing skills. The exercise in Slow Deep Breathing was embarrasing to several instructors.

Based on these findings, the workshop procedures and content were modified slightly. It was decided to retain the Slow Deep Breathing exercise because it was deemed particularly important for instructors to understand how this technique worked so that they would be in a position to recommend it to students experiencing test anxiety or having difficulty becoming relaxed while studying.

The operational tryout of the instructor workshops was conducted two weeks after the small group tryout. Forty-seven instructors initially volunteered to participate in the workshops and four sections were created, with eleven instructors assigned to each section. Due to summer leave and various job responsibilities, only half of these individuals actually participated: eight instructors each from the Precision Measuring Equipment (PME) and Inventory Management (IM) courses and seven instructors from the WM course.

Results and Conclusions

Evaluation data for the operational tryouts of the instructor workshops were provided by (1) instructor critiques of the workshop content and procedures; (2) instructors' subsequent use of their new skills in the assignment of students to particular study skills materials as recorded by the AIS data base; and (3) students' course performance after study skills remediation.

The results from the instructor critique indicated that instructors liked the workshop sessions and believed that they were very useful. It also appeared that the revisions made following the small group tryout were appropriate as all of the workshop activities were rated highly by a majority of the instructors. Additionally, anecdotal information indicated that instructors were enthusiastic about their new role. They returned to their respective courses and told other instructors about the workshops and presently personnel in several of the courses are attempting to schedule additional sessions of these workshops. In talking with instructors who participated in the workshops, it was learned that this type of training was what they had been looking for since the implementation of the AIS. Instructors explained that it was not that they disliked the system but that they had no idea how to adapt their skills to CMI requirements.

In order to assess instructor's use of their new skills and students' course performance after remediation, data were collected for ten weeks following the instructor workshops. Data on these two evaluation questions were provided by instructors who participated in either the small group or operational tryout of the workshops. During this time, an average of more than two students per instructor were given study skills module(s) assignments. Judging by discussion with instructors and data clerks, however, this statistic may be misleading because substantially more study skills assignments were made but not recorded in the AIS data base. It was concluded, therefore, that the lack of substantial data was due to a weakness in the workshop's description of the procedures for recording these data and not due to instructors' dislike or indifference to their new skills and materials.

With respect to student course performance following study skills remediation, the data show dramatic student improvement in block times and scores (see Table 1). These data are shown as standardized scores to allow comparisons of performance changes across courses with variable block lengths and end-of-block test difficulties. As can be seen, all five IM students and four of the six WM students improved their block times from pre- to post-study skills remediation and this remediation led to an overall improvement in block times across the two courses of more than one standard deviation (from 2.54 to 1.27).

Pre- and post-block score changes for students in the two courses indicate that all five IM students and five of the six WM students improved their block score performance following study skills remediation. The overall improvement in student block scores was again more than one standard deviation (from -1.19 to .03).

Of those students administered study skills remediation, only two IM students had data on both the initial and end-of-course administrations of the SSQ. These students' data on the total scale (SSQUES) and four subscales (READCM, MEMORY, TSTTAK, CONMGT) are shown in Table 2 along with a listing of the particular Study Skills Modules these students were assigned during the course. It is interesting to note that the student who received all four modules demonstrated the greatest improvement in his assessment of his study skills, whereas there was little change in the ratings of the student who received only the Reading Comprehension Module.

A subsequent evaluation of the SSQ indicated that not only do the questionnaire and its subscales demonstrate good reliability and construct validity, but results of predictive validity analyses indicate that the test reliably discriminates between students who perform satisfactorily versus poorly in the AIS CMI environment. Such discrimination was reliable for block times and scores and lesson times and scores, suggesting the sensitivity of the questionnaire to different CMI criterion variables.

Educational Importance of the Study

This project has demonstrated the positive benefit of instructor and student skill training in a CMI system for reducing the costs of military technical training by helping instructors and students become more efficient and effective in their respective roles. Project results also indicate that implementation problems caused by instructor or staff attitude and motivation can be alleviated by addressing the transition concerns of these instructional personnel and providing them with concrete and utilitarian skills appropriate to their new responsibilities. An important aspect of this instructor training seems to be the opportunity it gives instructional personnel to discuss and share their uncertainty about their new role in a computer-based instructional environment.

References

Arp, L., & Semb, G. An analysis of the use of student proctors in a personalized college business course. *Journal of Personalized Instruction*, 1977, *2*(2), 92-95.

Bunderson, C. V. *Instructional design, computers, and teacher education.* Technical Memo No. 2, University of Texas at Austin, Computer-Assisted Instruction Lab, December 1970.

Cadenhead, K. A brief report on personalized teacher education. *Journal of Personalized Instruction*, 1976, *1*(1), 51-52.

Campbell, D. F. New roles for occupational instructors. Paper presented at the Texas Post-Secondary Office Occupations Workshop, Community College of the Air Force, Lackland AFB, TX. May 1977.

Cohen, M. W., Emrich, A. M., & deCharms, R. Training teachers to enhance personal causation in students. *Interchange*, 1976-77, *7*(1), 34-39.

deCharms, R. *Enhancing motivation: Change in the classroom.* New York: Irvington, 1976.

Dyer, C. A. *Preparing for computer assisted instruction.* Englewood Cliffs, NJ: Educational Technology Publications, 1972.

Falzetta, J. N. Computerization--a key to humanization. *AEDS Journal*, 1973, *7*(1), 2-10.

TABLE 1

Standardized Pre/Post Study Skills Remediation
Block Times and Scores for IM and WM Students

Course/ Student	Pre-Treatment Score	Post-Treatment Score	Modules Taken*
Block Time			
IM-1	1.77	1.57	RC, M, CM
IM-2	1.94	.09	RC, M, CM
IM-3	2.75	.89	RC
IM-4	5.45	2.00	RC, M, TT, CM
IM-5	2.40	1.96	TT
IM Average	2.86	1.30	
WM-1	2.48	-.84	RC, M, TT, CM
WM-2	2.34	1.40	RC, M, TT, CM
WM-3	1.89	1.22	RC, M, TT, CM
WM-4	4.51	2.60	RC, M, TT, CM
WM-5	.71	1.42	RC
WM-6	1.30	1.63	RC
WM Average	2.21	1.24	
Overall Average	2.54	1.27	
Block Score			
IM-1	-1.12	-.13	RC, M, CM
IM-2	-1.14	-.59	RC, M, CM
IM-3	-1.88	-.93	RC
IM-4	-2.24	-.29	RC, M, TT, CM
IM-5	- .89	.25	TT
IM Average	-1.45	-.34	
**WM-1			
WM-2	-1.82	.95	RC, M, TT, CM
WM-3	- .11	.30	RC, M, TT, CM
WM-4	-2.43	-.32	RC, M, TT, CM
WM-5	- .20	.51	RC
WM-6	- .10	.61	RC
WM Average	- .93	.41	
Overall Average	-1.19	.03	

* RC = Reading Comprehension; M = Memorization; TT = Test Taking;
CM = Concentration Management
** No Pre/Post Score Data Available

TABLE 2
Changes in Student Scores on Initial and End-of-Course Study Skills Questionnaire Administrations

COURSE/ STUDENT	SCALE	INITIAL SCORE	END SCORE	MODULE(S) TAKEN
IM-1	SSQUES	69	70	Reading Comprehension
	READCM	19	21	
	MEMORY	18	17	
	TSTTAK	12	15	
	CONMGT	20	17	
IM-2	SSQUES	52	68	All 4 Modules
	READCM	24	21	
	MEMORY	10	20	
	TSTTAK	02	16	
	CONMGT	16	11	

Hansen, D. N., & Harvey, W. L. Impact of CAI on classroom teachers. Educational Technology, 1970, 10(2), 46-48.

Harvey, W. L. Learning systems for the future. Journal of Educational Technology Systems, 1977-78, 6(1), 3-6.

Hursh, D. E. Personalized systems of instruction: What do the data indicate? Journal of Personalized Instruction, 1976, 1(2), 91-105.

Kool, B., & Geddes, C. The teacher's role in computer assisted instructional management. Educational Technology, 1970, 10(2), 42-45.

Lamos, J. P. The instructor is essential. USAF Instructors Journal, 1971, 8(2), 55-59.

McCombs, B. L., Dobrovolny, J. L., & Judd, W. A. Development and evaluation of student skill modules to reduce training time in computer-managed instruction. AFHRL Final Technical Report, DARPA Contract No. MDA903 77 C 0144, in press.

Nash, A. M., & Grimm, R. L. Individualized instruction in mathematics in an open junior high school. *Journal of Personalized Instruction*, 1976, 1(1), 23-27.

Stanchfield, J. M. The teacher's role in educational technology. Paper presented at the Annual Meeting of the International Reading Association, New Orleans, May 1974.

Wilson, H. A. Preparation of school personnel for the use of instructional technology. Paper presented at the Annual Meeting of the International Reading Association, Denver, May 1973.

Wittrock, M. C., & Lumsdaine, A. A. Instructional psychology. *Annual Review of Psychology*, 1977. Pp. 417-459.

Instructional Styles for Microcomputer-Based Educational Materials

Chaired by Harold J. Peters
CONDUIT/University of Iowa

The proliferation of stand-alone microcomputers poses new opportunities and challenges for developers of computer-based educational materials. The opportunities appear in the form of inexpensive input/output features with impressive capabilities and growing user acceptance of microcomputers as common, everyday items. The challenges posed are to make really effective use of the new capabilities, and in so doing to approach the goal of producing materials that are friendly to the user, attractive and motivating to use, and in the end educationally effective.

Based on their experiences with various stand-alone systems at their own institutions and exchanges with many other developers nationally, the following speakers will comment on the implications of microcomputers for authors.

Developing and Translating Instructional Materials on the APPLE-II for State-Wide Use

Kevin Hausmann, Minnesota Educational Computing Consortium

Implementing Instructional Materials on IBM 5100 Systems; User Acceptance

David A. Smith, Duke University

Using PILOT for Development of Microcomputer-Based Instructional Materials

George Gerhold, Western Washington State University

Approaches for Producing Graphics-Oriented Materials for Transport to Various Microcomputer-Based Systems

Alfred Bork, University of California-Irvine

Speculations on Future Developments; Implications for Today's Authors

Karl Zinn, University of Michigan

AUTHOR INDEX

Arnold Arons 130
Tryg A. Ager 112,335
Bill Arms 99

Daniel E. Bailey 2
William L. Ballenger 274
Michael A. Barletta 354
Michael D. Barr 227
Harold Benenson 221
Warren Bickel 285
Alfred Bork 112,113,130,386
Kenneth Bowles 84
Lia Brillhart 21
Bobby Brown 113

Lowell A. Carmony 44
W.J. Carroll 378
A. Rees Clark 107
Harvey A. Cohen 170
Francis J. Colella 194
Ronald W. Collins 114
S.L. Cooke, Jr. 347
John Patrick Cronin 280
R.S. Cunningham 291

Cecilia Daly 150
Henry W. Decker 326
J. Richard Dennis 26,31
Terry L. Dennis 194
Con Dietz 161
Jacqueline Lea Dobrovolny 380
Steven J. Duff 114

James E. Eilers 86
Lynda B.M. Ellis 311
David W. Embley 150
Gerald L. Engel 1
Kathryn Erat 316

Richard Fairley 261

Alfredo García 371
Laël C. Gatewood 311
Edward J. Gauss 60
George Gerhold 386

Keith A. Hall 7
John W. Hamblen 36,39,65
Carol Hargan 164
Kevin Hausmann 386
Margret Hazen 150
Harry Hedges 84
James G. Henkel 360
Jim Hobbs 205
James F. Hornig 100
Richard E. Horton 268
Robert L. Horton 197
Elisha Huggins 100
Herman D. Hughes 141
Beverly Hunter 164

Edwin J. Jaufmann, Jr. 354
G. Christian Jernstedt 100
James A.E. John 18
James W. Johnson 14,160
K. Jeffrey Johnson 92
Clinton Jones 65
Gary L. Jones 301
Bhairav D. Joshi 86
Janene Judd 112
Wilson A. Judd 380
Steven Just 221

Steven I. Kartashev 262
Svetlana P. Kartashev 262
Kent T. Kehrberg 343
Kenneth D. Keith 301
Gene A. Kemper 161
Rosemary N. Killam 329
Ronald King 26
Daniel Klassen 26

J.A.N. Lee 135
Leo Leveridge 84
Jesse Lewis 65
G. Jack Lipovski 262
Laurette F. Lipson 301
Elizabeth R. Little 160

Ken Magel 261
Michael Maner 48
Sr. Patricia Marshall 65
William R. Martin 347
Ted Martz 112
Bodie Marx 160
Barbara L. McCombs 380
James McDonald 335
Robert J. McGlinn 44
Susan McGrath 316
J.A. Metzler 378
Stuart D. Milner 26,27
David Moursund 26

George Nagy 150
G.V. Nelson 378

Robert Oberg 297

Donald W. Paden 227
Samuel Patton 84
Harold J. Peters 386
Richard E. Pogue 160
James L. Poirot 26,36,39
Gerald J. Porter 237
James D. Powell 36,39
Mark Pozefsky 201
William Prange 150
Kenneth Preskenis 297

C.V. Ramamoorthy 262
Douglas Ray 329

Retnadhas 262
Thomas Rice 326
Jon Todd Rickman 205
Peter J. Rizza, Jr. 280
Thomas Roos 99
Sara Rude 371
Martin Rudnick 271

Robert Schenk 179
K.W. Scholz 264
Douglas Shawan 21
Edmund Skellings 85
D.E. Sloane 264
David A. Smith 386
Cynthia J. Solomon 250
Steven Stack 211
William J. Stevenson 194
Gary W. Stoaks 52
Mary Jane Sunkel 205
George F. Sutherland 18
Kathleen M. Swigger 48

Elliot A. Tanis 233
Robert P. Taylor 26,36,39
Alan L. Tharp 201
Ron Thorkildsen 285
Kenneth J. Thurber 262

Zvonko G. Vranesic 101

Daniel H. Watt 255
Robert T. Ward 271
Frank W. Werner 189
Linda Wilkinson 360
Joseph G. Williams 285
Robert L. Williams 354

Karl Zinn 386